1990

Numerical investigation of the buoyancy effects due to the density extremum in water.

Venugopal. Subramanyam
University of Windsor

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

NUMERICAL INVESTIGATION OF THE BUOYANCY EFFECTS DUE TO THE DENSITY EXTREMUM IN WATER

by
Venugopal Subramanyam

A thesis
submitted to the
Faculty of Graduate Studies and Research
through the Department of
Mechanical Engineering in partial fulfillment
of the requirements for the Degree of
Master of Applied Science at the
University of Windsor

Windsor, Ontario, Canada
1990

ISBN 0-315-61866-3

To Amma, Mambu and Sri Satya Sai Baba

ABSTRACT

Numerical studies have been conducted for the flow in a differentially heated square cavity and the results compared with those of Lin and Nansteel. A non-uniform grid algorithm has been developed and written in Fortran and numerical studies conducted with the differentially heated square cavity. The above developed scheme has been validated by investigating the flow over a flat plate . The program was then used to investigate the flow over an horizontal ice sheet and the results have been presented.

ACKNOWLEDGEMENTS

I wish to express my profound appreciation and gratitude to Dr.N.W.Wilson for his encouragement, insight and patience throughout the course of the study.

Special thanks to Dr.Sridhar and Dr.Rankin for their encouragement and help as professors and as very caring individuals who motivated me when I needed it most.

Thanks are also due to Dr.Zamani and Dr.Kierkus for their insight during the course of the study. Thanks go to Ms. Jayalakshmi Sreedharan of the computing centre for her suggestions in resolving the many difficulties encountered.

This endeavour reflects the support I received from home, for which , I thank my mother and brothers .

This study was financially supported by the Natural Sciences and Engineering Research Council of Canada through grant OGP0001243 / A1243 .

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

NOMENCLATURE

g	gravitational acceleration, m/sec^2
i	grid coordinate in the x direction
j	grid coordinate in the y direction
k	thermal conductivity, $W/(m^0 c)$
k_i	x coordinate node corresponding to the start of the ice sheet
k_e	x coordinate node corresponding to the end of the ice sheet
L	length of the square enclosure and also the length of the domain for flow over an ice sheet, m
in	no. of grids in the x direction
jn	no. of grids in the y direction
$Nu(x,y)$	local Nusselt number, equation 2.16
$\overline{Nu}(x)$	vertically averaged Nusselt number, equation 2.18
$\bar{p}$	modified pressure , $\bar{p}' + \rho_c g \bar{y}$, N / m^2
$\bar{p}'$	pressure, N / m^2
Pr	Prandtl number
q	exponent in density-temperature relationship, equation 2.7
q''	heat flux, W / m^2
R	density distribution parameter, equation 1.1
Ra	Rayleigh number, equation 2.10
$\bar{T}$	temperature, K
t	dimensionless time
$\bar{t}$	time, sec
u	dimensionless horizontal velocity
$\bar{u}$	horizontal component of velocity, m/sec
v	dimensionless vertical velocity
$\bar{v}$	vertical component of velocity, m/sec
x	dimensionless horizontal coordinate
$\bar{x}$	horizontal coordinate, m
Δx	horizontal grid spacing
y	dimensionless vertical coordinate
$\bar{y}$	vertical coordinate, m
Δy	vertical grid spacing

GREEK SYMBOLS :

α	thermal diffusivity, m^2/ s
α_1	constant in density temperature relationship, equation 2.7
β	coefficient of thermal expansion, K^{-1}
μ	dynamic viscosity, Kg/ m s

ν	kinematic viscosity, m^2/s
ω	dimensionless vorticity
$\bar{\omega}$	vorticity, sec^{-1}
ρ	density, m/sec
ϕ	$\dfrac{\bar{T} - \bar{T}_c}{\bar{T}_h - \bar{T}_c}$
ψ	dimensionless stream function
$\bar{\psi}$	stream function, Kg/ m s

SUBSCRIPTS

c	cold wall
h	hot wall

CHAPTER 1

THE BUOYANCY INDUCED FLOWS IN WATER

1.1 Introduction

The phenomenon of the natural convection of water near its density extremum is of primary importance in the study of the formation and dissipation of ice. The buoyancy induced flow of water in a cavity with differentially heated walls can result in several interesting multicellular flow situations .

Water has a density extrema at about $4^{\circ}C$. As a result, if a vertical ice sheet were in contact with water near its density extremum, the temperature differences in the body of water will result in buoyancy induced flows. If the water near the ice sheet has a lesser density than the water far away from the ice sheet, the lighter body of water rises up and the denser water comes down, resulting in a dominant upflow of water near the ice sheet. On the other hand, if the water near the ice sheet is denser than the water away from it, there is going to be a dominant downflow of water near the ice sheet. There is also a possibility of water being densest in the region between the portions close to and farther away from the ice sheet, and this can result in an unstable flow condition in the water.

Instead of the vertical ice wall in a quiescent medium, if an isothermal wall at 0^{0} C forms a vertical wall in a square cavity, where the other vertical wall is maintained at a higher temperature, the upflows and downflows near the 0^{0} C wall results in water from the unaffected regions of the cavity moving towards this region of upflows and downflows and this could lead to several

interesting flow situations.

The other interesting situation involving buoyancy effects occurs in flow above an ice sheet, where cross stream buoyancy effects play an important role in the flow phenomenon. The buoyancy force acts upwards, against the direction of the gravitational forces, and a balance between the inertial, viscous and buoyancy forces can lead to some unique flow situations.

Most of the work involving flow near a vertical isothermal wall in a quiescent medium was done using the boundary layer approximations [6, pp 407 - 415] and these were basically aimed at getting some form of similarity solutions for the above phenomenon. The mathematical treatment becomes extremely complicated with the inclusion of melting on the ice surface and so most of the work is done assuming isothermal walls . Furthermore, the mathematical treatments failed totally in the regions of convective inversions of water and there was no good scheme of comparison with the experimental results (Gebhart and Mollendorf ,1976) [6, pp 415 - 425].

The experiments, on the other hand, had practical difficulties on their side, in their inability to put forth ideal conditions for studying the various flow phenomenon. Sequential photographs (V.P.Carey and Mollendorf, 1981) [6, pp 427 - 431] fail to yield a comprehensive nature of the flow phenomenon, and a study of several different cases experimentally, is close to impossible.

Hence, it was decided to go in for a numerical scheme where a good combination of the advantages of both the mathematical and experimental approaches could be expected, in the sense of being able to deal the flow situations without having to go in for boundary

layer approximations and the flexibilty of running a variety of cases.

The two most popular numerical schemes are the finite element and the finite difference schemes. The finite element scheme consists of dividing the computational domain into a finite number of elements, writing nodal equations which aptly describe the flow situation for these elements, assembling these equations with the appropriate boundary conditions into a banded matrix and solving these to get the values of the variables at these nodes. This technique is primarily used in the field of stress analysis in solids. The use of finite element methods for convective heat transfer in fluid flow, which involves solving several equations simultaneously, is not recommended.

The finite difference technique consists of dividing the computational domain into a finite nodal mesh and writing the partial differential equations which describe the phenomenon in the form of finite difference equations at each of the nodes. These finite difference equations are assembled as matrices and solved by a simple or higher form of back substitution, which is a solution procedure for the matrices involved. For the finite difference scheme , the equations are to be solved simultaneously, and the scheme was carried on until a steady state situation was attained. Steady state, in the finite difference sense, is defined as that state for which subsequent iterations do not produce appreciable changes in the values of the nodal variables. Also, the mesh system chosen should aptly describe the flow situation , only a solution for the finite difference equations and not the partial differential equation is obtained and this model should be checked for grid dependencies. This was done by

choosing a finer mesh , thereby approaching the differential equation from the difference equation. Furthermore, since the flow phenomenon is modelled as a time dependent process , a finite time step has to be chosen as the initial approximation and this time step has to be checked out by going for a lower time step. Because of the complexities involved in the formulation and the subsequent testing of the numerical scheme, the finite difference numerical scheme that was developed was tested against an earlier published result for a differentially heated square cavity.

Furthermore, the simple case of a laminar driven cavity was modelled using the above numerical scheme and the validity of the numerical procedure was reinforced. Finally the flow of cold water over an ice sheet was modelled based on this scheme and the thesis is a detailed documentation of the numerical procedures, the assosiated instabilities and the convergence criteria . The physical flow phenomenon as predicted by the numerical scheme is also discussed in detail.

1.2 Literature Survey

Natural convection of water near its density extremum has been investigated for several years. Because of the anomalous behaviour of water, flow situations occuring near the 4°C region are quite complex. Buoyancy force reversals can result in the total reversal of the dominant flow direction and this phenomenon is called as convective inversion.

Codegone (1939) [6, p 406] was the first to demonstrate convective inversion due to the density extremum for water in contact

with a vertical ice sheet .

Merck (1953) [6, p 406] was the first to investigate such flows with water in contact with a vertical ice sheet. He used an integral method for predicting the local heat transfer for ice melting in fresh water . Convective inversion was predicted to occur at about 5.3°C and the heat transfer rate was found to undergo a minimum at that temperature.

Oborin (1967) [6, p 407] studied the heat transfer from spheres and horizontal cylinders in cold water and his experimental results were in good conformity with the earlier predictions of Merk. Schenk and Schnekels (1963) [6, p 407] measured the heat transfer in an ice sphere melting in cold water and these results agreed well with the predictions of Merk.

Bendell and Gebhart (1976) [6, p 423] measured the melting rates of vertical ice slabs in ambient water at temperatures between 2°C and 20°C. They used thermocouples to measure the temperatures near the ice sheets and deduced the flow direction from this information; however, these results were later found to be erroneous.

Gebhart and Mollendorf (1978) [6, pp 415 - 427] performed a similarity analysis for the relevant equations for a vertical isothermal surface in a quiescent medium and they were able to get converged solutions in the range $R < 0$ and $R > 0.5$, where R is given by

$$R = \frac{T_m - T_\infty}{T_0 - T_\infty} \qquad (1.1)$$

T_m is the temperature at which water exhibits a density extremum, this being 4.029325^0C. T_∞ is the free stream temperature and T_0 is the surface temperature of the isothermal wall.

If T_0 is 0^0C for an ice surface, the values of R between 0 and 0.5 corresponds to free stream water temperatures between 4°C and 8°C approximately. In the range of $0. < R < 0.5$, converged solutions were not obtainable and this was attributed to the first order boundary layer approximations used in their derivations.

Wilson and Vyas [19] performed an experimental investigation on the velocity profiles due to free convection of a vertical ice sheet immersed in fresh water at temperatures between 2°C and 7°C. They found that when the temperature of water was less than 4°C , there was an entirely upward flow and when the temperature was greater than 7°C, there was an entirely downward flow. Between these two temperatures , the flow was unsteady and fluctuated with time.

Wilson and Lee [17] developed a two dimensional steady state finite difference program based on the methods of Gosman et al. [7], and modelled the surface of a vertical ice wall immersed in water at various temperatures and they too were not able to obtain converged solutions in the water temperature range of 4.5°C and 5.7°C. They suggested that the flow may be oscillatory in that range and recommended a transient analysis for the same.

V.P.Carey and B.Gebhart [1] did an experimental investigation on the melting of an ice sheet completely immersed in cold water at the temperature ranges of 3.9°C and 8.4°C. They

observed that the flow conditions pertaining to R ranges of 0.15 to 0.29 are not of the boundary layer type and that the flow was unsteady in this range.

Gebhart et al., [6, p 463] visualised the flow under a rectangular ice slab with the ambient water temperatures ranging from - 1.75°C and 3.0°C. They observed that the principal flow regime was a downflow near the centre of the ice sheet and that the flow split and spread into a horizontal layer.

Lin and Nansteel [11] developed a computer program based on the alternating direction implicit (ADI) scheme for observing the flow situation of water near its density extremum in a square two dimensional cavity.

From the above studies , it was concluded that a numerical investigation was appropriate because of the flexibility in dealing with a variety of cases , which was the drawback for the experimental investigations and not having to use the boundary layer approximations, which was the drawback for the mathematical treatments.

1.3 Lin - Nansteel Algorithm

Wilson and Sarma [18] performed a numerical investigation on the flow of cold water over an horizontal ice sheet based on the methods of Gosman et al., [7] . They obtained solutions similar to that of forced convection solution for high free stream velocities and low free stream velocities resulted in oscillatory solutions. Their work was inconclusive in establishing whether these oscillatory solutions were the result of numerical instabilities or were an

attempt by a steady state methodology to describe a flow phenomenon which is actually oscillatory in nature.

Investigation of this problem forms a major portion of the present work. Since the boundary conditions for this flow situation can lead to instabilities, flows that can have simpler boundary conditions were modelled, so that the validity of the numerical scheme could be tested.

Hence, the most promising algorithm in the literature was tried out, and the computer code for the Lin - Nansteel alternating direction implicit (ADI) scheme was developed.

1.4 OBJECTIVES

The objectives were defined at this stage and they were as follows :

(1) To use the ADI scheme for the differentially heated square cavity to gain familiarity with the numerical code and the boundary conditions and to establish the validity of the scheme .

(2) To develop the computer code for the non-uniform scheme and test it for the differentially heated square cavity.

(3) To develop the computer code for the case of flow of cold water over an ice sheet .

(4) To use the computer code developed in (3) to study the physical behaviour of the flow situation at low flow velocities , when the buoyancy effects become dominant to look into the possibility of oscillatory behaviour as concluded by Wilson and Sarma [18] .

Having thus put forth the objectives, the mathematical treatment of the flow situation is dealt with in the next chapter.

CHAPTER 2

MATHEMATICAL FORMULATION

2.1 The ADI Scheme Based on Lin and Nansteel's Work

The differentially heated square cavity is shown in Sketch 1. The length of the cavity is $\bar{L}$, and the hot and cold wall temperatures are indicated by $\bar{T}_h$ and $\bar{T}_c$ respectively and the horizontal walls are insulated.

The following assumptions were made to simplify the analysis . The flow is laminar, two dimensional and incompressible. Physical properties such as the kinematic viscosity , specific heat capacity , thermal conductivity and Prandtl number are constant. Reference density was choosen as that corresponding to 0^0C, the reference temperature. All fluid properties were evaluated at the reference temperature. The density was also assumed constant except in the buoyancy terms in the vertical momentum equation , where the density was assumed to vary with temperature only. This assumption is the first part of the Boussinesq approximation.

The governing equations are the continuity equation 2.1, the momentum equations 2.2 and 2.3 and the energy equation 2.4.

$$\frac{\partial \bar{u}}{\partial \bar{x}} + \frac{\partial \bar{v}}{\partial \bar{y}} = 0 \qquad (2.1)$$

$$\frac{D\bar{u}}{D\bar{t}} = -\frac{1}{\rho_c}\frac{d\bar{p}'}{d\bar{x}} + \nu \nabla^2 \bar{u} \qquad (2.2)$$

$$\frac{D\bar{v}}{D\bar{t}} = -\frac{1}{\rho_c}\frac{d\bar{p}'}{d\bar{y}} + \nu \nabla^2 \bar{v} - \frac{\rho}{\rho_c} g \qquad (2.3)$$

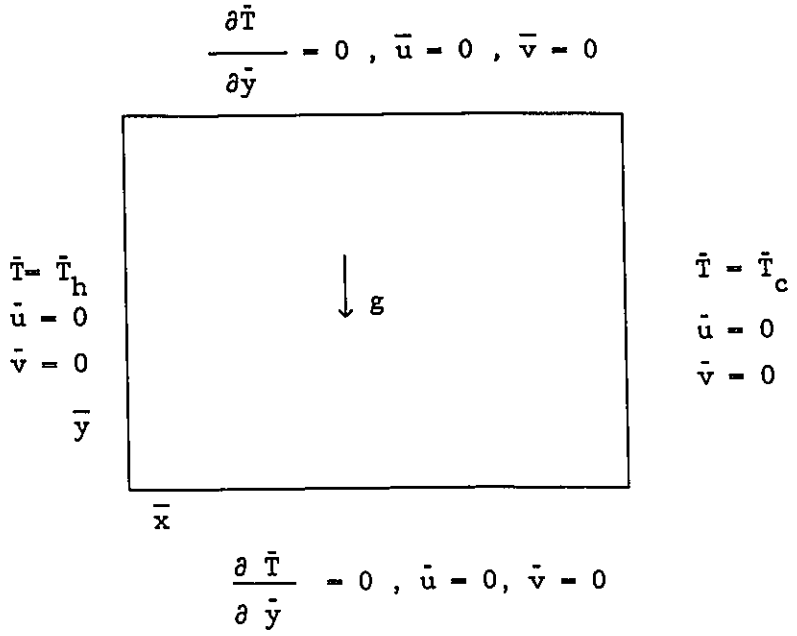
$$\frac{D\bar{T}}{D\bar{t}} = \alpha \nabla^2 \bar{T} \qquad (2.4)$$

Introducing $\bar{p}$ as the modified (piezometric) pressure, we have

$$\bar{p} = \bar{p}' + \rho_c g y$$

Hence,

$$\frac{d\bar{p}'}{d\bar{y}} = \frac{d\bar{p}}{d\bar{y}} - g\rho_c \qquad (2.5)$$



Sketch 1 : Differentially heated square cavity

The boundary conditions are

At $\bar{t} = 0$,

$$\bar{u} = 0 \ , \ \bar{v} = 0$$

$$\bar{T} = \frac{\bar{T}_h + \bar{T}_c}{2}$$

The velocities are set as zero everywhere because the fluid is at rest initially. The mean temperature between the hot and cold walls was taken as the temperature of the fluid initially.

At the instant $\bar{t} > 0$, the left vertical wall is kept at the hot wall temperature, which varies according to the value of R and the right vertical wall temperature is set at the cold wall temperature, which is always at 0^0C . The horizontal walls are insulated and hence the temperature gradients in the $\bar{y}$ direction at these walls is set as zero. The $\bar{u}$ and the $\bar{v}$ velocities along the walls are set to zero in compliance with the no slip boundary condition.

$$\bar{t} > 0$$

$$\bar{T}(0,\bar{y}) = \bar{T}_h$$

$$\bar{T}(\bar{L},\bar{y}) = \bar{T}_c \qquad (2.6)$$

$$\frac{\partial \bar{T}\ (\ \bar{x}\ ,\ 0)}{\partial \bar{y}} = \frac{\partial \bar{T}\ (\bar{x}\ ,\ \bar{L})}{\partial \bar{y}} = 0$$

$\bar{u} = \bar{v} = 0$ on the boundary

Since the second part of the Boussinesq approximation, namely the linear variation of density with temperature is not valid for water near its density extremum, a precise and concise relationship for describing the density had to be chosen. From the

many correlations that represent the density of water as a function of temperature , that due to Gebhart and Mollendorf (1977) [6, p 401] was chosen due to its simplicity and accuracy.

Their relation is

$$\rho = \rho_m \left[1 - \alpha_1 \left| \bar{T} - \bar{T}_m \right|^q \right] \qquad (2.7)$$

Where

$\rho_m = 999.972 \text{ kg/m}^3$

$\alpha_1 = 9.297173 \text{ e-06 } (^\circ C)^{-q}$

$\bar{T}m = 4.029325 \ ^\circ C$

$q = 1.894816$

Introducing 2.5 & 2.7 , to represent the density differences in terms of the temperature differences, 2.2 & 2.3 become

$$\frac{D\bar{u}}{D\bar{t}} = -\frac{1}{\rho_c}\frac{\partial \bar{p}}{\partial \bar{x}} + \nu \nabla^2 \bar{u} \qquad (2.8)$$

$$\frac{D\bar{v}}{D\bar{t}} = -\frac{1}{\rho_c}\frac{\partial \bar{p}}{\partial \bar{y}} + \nu \nabla^2 \bar{v} + \frac{g\alpha_1 \rho_m}{\rho_c}\left[\left| \bar{T} - \bar{T}_m \right|^q - \left| \bar{T}_c - \bar{T}_m \right|^q \right] \qquad (2.9)$$

The stream function and vorticity were defined and the momentum equations were cross differentiated to eliminate the pressure term and a single vorticity transport equation was obtained as opposed to the two momentum equations. Thus,

$$\frac{\partial \bar{\psi}}{\partial \bar{y}} = \bar{u} \quad , \quad \frac{\partial \bar{\psi}}{\partial \bar{x}} = -\bar{v} \quad \text{AND} \quad \bar{\omega} = -\nabla^2 \bar{\psi} \qquad (2.10)$$

The variables such as the length, velocity and temperature are rendered dimensionless so that the basic equations could be represented in terms of the dimensionless numbers such as Ra and Pr. Hence,

$$x = \frac{\bar{x}}{L} \quad , \quad y = \frac{\bar{y}}{L} \quad , \quad u = \frac{\bar{u}\, L}{\nu} \quad , \quad v = \frac{\bar{v}\, L}{\nu}$$

$$\phi = \frac{\bar{T} - \bar{T}_c}{\bar{T}_h - \bar{T}_c}$$

$$\psi = \frac{\bar{\psi}}{\nu} \quad , \quad \omega = \frac{\bar{\omega}\, L^2}{\nu^2} \quad , \quad t = \frac{\bar{t}\, \nu}{L^2}$$

$$Ra = \frac{g\, \alpha_1 \rho_m L^3 \left(\bar{T}_h - \bar{T}_c \right)^q}{\rho_c\, \nu\, \alpha}$$

$$Pr = \frac{\nu}{\alpha} \qquad R = \frac{\bar{T}_m - \bar{T}_c}{\bar{T}_h - \bar{T}_c}$$

The governing equations now are the vorticity transport equation (2.11) , the energy equation (2.12) and the stream function equation (2.13) .

$$\frac{D\, \omega}{D\, t} = \frac{Ra}{pr}\, q \left| \phi - R \right|^{q-2} (\phi - R) \frac{\partial \phi}{\partial x} + \nabla^2 \omega \qquad (2.11)$$

$$\frac{D\, \phi}{D\, t} = \frac{1}{Pr}\, \nabla^2 \phi \qquad (2.12)$$

$$\nabla^2 \psi = - \omega \qquad (2.13)$$

At t = 0, the dimensionless u and v velocities are zero

on the walls and the dimensionless temperature is kept at 0.5, which is the mean temperature between the hot and cold walls. Thus,

$$u = v = 0 \; ; \quad \phi = 1/2$$

At $t > 0$, the hot and cold walls are maintained at the dimensionless temperatures of 1.0 and 0.0 and the stream function is set as zero all along the boundary in compliance with the no slip boundary condition.

$$\phi(0,y) = 1.0 \; ; \; \phi(1,y) = 0.$$

$$\frac{\partial \phi\ (x,0)}{\partial y} = \frac{\partial \phi\ (x,1)}{\partial y} = 0.$$

$$\psi = \frac{\partial \psi}{\partial x} = \frac{\partial \psi}{\partial y} = 0, \quad \text{on the boundary.}$$

Equation 2.13 for the stream function is modified by introducing the false time step as

$$\frac{\partial\ \psi}{\partial\ t} = \nabla^2 \psi + \omega \qquad (2.14)$$

The above modification is performed because of the advantage of using the alternating direction implicit scheme for the above equation.

In the above equation, $\frac{\partial\ \psi}{\partial\ t}$ is the false transient stream function equation which modifies equation 2.13.

The equations 2.11, 2.12 and 2.14 can be cast into the following general form as

$$\frac{\partial\ \xi}{\partial\ t} = - a \frac{\partial\ \xi}{\partial\ x} - b \frac{\partial\ \xi}{\partial\ y} + c \nabla^2 \xi + d \qquad (2.15)$$

The individual equations are identified in the table 2.1 by means of the corresponding values of the coefficients.

The above generalisation is made so that the numerical approach could be explained in a systematic manner in the following chapter.

Before going on to the next chapter , the mathematical approach for the calculation of the heat transfer by means of the Nusselt number will be discussed .

ξ	a	b	c	d
ω	u	v	1	$\frac{Ra}{Pr} q \mid \phi - R \mid^{q-2} (\phi - R) \frac{\partial \phi}{\partial x}$
ϕ	u	v	1/Pr	1
ψ	0	0	1	1

Table 2.1 Table of coefficients for equation 2.15

2.2 NUSSELT NUMBER CALCULATION

The information related to the heat transfer is best expressed in terms of the Nusselt number, which is a dimensionless representation of the heat transfer in the square cavity.

The hot vertical walls lose their heat by natural convection to the water inside the cavity, this heat being transferred to the cold wall. There is no heat loss through the horizontal walls as they are insulated.

The Nusselt number (Nu) is defined as follows.

$$Nu(x,y) = \frac{q'' \bar{L}}{k (\bar{T}_h - \bar{T}_c)} \qquad (2.16)$$

Where q'' is the heat transfer per unit area of surface .

Since the heat transfer at the walls , which have no slip is only by conduction, the Nusselt number could be expressed as

$$Nu(x,y) = - \left. \frac{\partial \phi}{\partial x} \right|_{x = 0} \text{ or } - \left. \frac{\partial \phi}{\partial x} \right|_{x = L} \qquad (2.17)$$

Hence the average Nusselt number at a vertical wall is obtained by integrating the local Nusselt number at a point over the height of the cavity.

$$\text{Hence, } \quad \overline{Nu}(x) = - \int_0^1 \frac{\partial \phi}{\partial x} \, dy \qquad (2.18)$$

Though the above expression gives the Nusselt number as a function of x, there is no variation of the Nusselt number along the x direction because of the insulated horizontal walls.

The heat transfer is thus represented by the Nusselt number and this concludes the mathematical formulation of the natural convection in a square cavity. In the next chapter, the numerical formulation for the square cavity and the subsequent results will be discussed.

CHAPTER 3

THE SQUARE CAVITY

3.1 The Numerical Approach

3.1.1 Introduction

The governing equations for energy and vorticity are parabolic , and the stream function equation is of the elliptic Poissonian type. These equations were solved by choosing a finite domain of computation and creating a finite difference mesh in the domain. There is a difference equation for every node and this results in a system of differential equations for every partial differential equation describing a particular parameter. Thus, we have three systems of difference equations for the vorticity, energy and stream function equations .

The above equations were formulated based on the Alternating Direction Implicit (ADI) scheme. The ADI methods were introduced in companion papers by Peacemen and Rachford (1955) and Douglas (1955) [12]. This method makes use of splitting of the time step to obtain an implicit method . The implicit method is the terminology which is applied to the technique for solving equations in which the value of a variable at the new time step appears on either side of the equation. The disadvantage of the schemes that are fully implicit is that the matrices that are obtained based on this scheme have entries in every row and column and hence the solution procedure is complicated. On the other hand, the ADI scheme reduces the system of difference equations , which are implicit, in a very convenient fashion so that the resulting matrix system has entries only in the main diagonal and off diagonal terms. This is called a tridiagonal

matrix , and the Thomas algorithm [12], which is a modified form of the back substitution technique, could be applied to get the solution for this matrix. The advantage of this method over the fully implicit methods is that each equation, though fully implicit, is only tridiagonal. The stability of this two dimensional method is unconditional , as in the fully implicit method.

The ADI scheme was used for the vorticity transport equation, the energy equation, and the stream function equation. The above equations were modified by introducing central differences for space coordinates and forward differences for the time steps. During the first half time step, the x directional variation was taken into account while the y directional variation was calculated at the previous time step and the reverse was applied to the second half time step.

The expression below gives the finite difference form of the equation 2.15 for the first half time step

3.1.2 First Half Time Step :

$$\frac{\xi_{i,j}^{n+1/2} - \xi_{i,j}^{n}}{(\Delta t / 2)} = - a_{i,j} \left\{ \frac{\xi_{i+1,j}^{n+1/2} - \xi_{i-1,j}^{n+1/2}}{2 \Delta x} \right\}$$

$$- b \frac{\partial \xi_{i,j}^{n}}{\partial y} + c \left\{ \frac{\xi_{i+1,j}^{n+1/2} + \xi_{i-1,j}^{n+1/2} - 2 \xi_{i,j}^{n+1/2}}{(\Delta x)^2} \right\}$$

$$+ \quad c \quad \frac{\partial^2 \xi^n_{i,j}}{\partial y^2} \quad + \quad d$$

(3.1)

Rearranging the above equations, to represent the entire equation in the form of coefficients for the values of the variable at nodes i+1, i and i-1 for the new time , we get

$$\left[\frac{a_{i,j}\,\Delta t}{4\,\Delta x} - \frac{c\,\Delta t}{2\,(\Delta x)^2} \right] \xi^{n+1/2}_{i+1,j}$$

$$+ \left[1 + \frac{c\,\Delta t}{(\Delta x)^2} \right] \xi^{n+1/2}_{i,j}$$

$$+ \left[-\frac{a_{i,j}\,\Delta t}{4\Delta x} - \frac{c\,\Delta t}{2(\Delta x)^2} \right] \xi^{n+1/2}_{i-1,j}$$

$$= \frac{\Delta t}{2} \left[\frac{\partial^2 \xi_{i,j}}{\partial y^2} - b_{i,j} \frac{\partial \xi_{i,j}}{\partial y} \right]$$

$$+ \frac{d\ \Delta t}{2} + \frac{\partial \xi^{n}_{i,j}}{\partial y}$$

(3.2)

In the above form of the equation the coefficients corresponding to the nodes i + 1, i and i - 1 as well as the right hand side coefficients were fed into the Thomas algorithm [12] subroutine that calculated these coefficients for all the nodes except the boundary and the values of the variable after the first half time step were found.

The same technique was applied to the second half time step with the y directional emphasis as explained below.

3.1.3 Second Half Time Step :

$$\frac{\xi^{n+1}_{i,j} - \xi^{n+1/2}_{i,j}}{(\Delta t / 2)} = - b_{i,j} \left\{ \frac{\xi^{n+1}_{i,j+1} - \xi^{n+1}_{i,j-1}}{2\ \Delta y} \right\}$$

$$- a \frac{\partial \xi^{n+1/2}_{i,j}}{\partial x} + c \left\{ \frac{\xi^{n+1}_{i,j+1} + \xi^{n+1}_{i,j-1} - 2\ \xi^{n+1}_{i,j}}{(\Delta y)^2} \right\}$$

$$+ c \frac{\partial^2 \xi^{n+1/2}_{i,j}}{(\Delta x)^2} + d$$

(3.3)

Rearranging the above equations as before to represent the entire equation in the form of the coefficients of the variables at the nodes j+1, j, and j-1 at the new time step,

$$\left[\frac{b_{i,j} \Delta t}{4 \Delta y} - \frac{c \Delta t}{2 (\Delta y)^2} \right] \xi^{n+1}_{i,j+1}$$

$$+ \left[1 + \frac{c \Delta t}{(\Delta y)^2} \right] \xi^{n+1}_{i,j}$$

$$+ \left[- \frac{b_{i,j} \Delta t}{4 \Delta y} - \frac{c \Delta t}{2 (\Delta y)^2} \right] \xi^{n+1}_{i,j-1}$$

$$= - \frac{\Delta t}{2} \left[\frac{\partial^2 \xi^{n+1/2}_{i,j}}{\partial x^2} - b_{i,j} \frac{\partial \xi^{n+1/2}_{i,j}}{\partial x} \right]$$

$$+ \frac{d \Delta t}{2} + \frac{\partial \xi^{n+1/2}_{i,j}}{\partial x}$$

(3.4)

In the above form of the equation the coefficients corresponding to the nodes j + 1, j and j - 1 as well as the right hand side coefficients were fed into the Thomas algorithm subroutine that calculated these coefficients for all the nodes except those at the boundaries and the values of the variable after the first time step were found.

By going through these two sets of computations, the values at the end of the whole time step was found by taking into account both the x and y directional variations and the computation could be carried on to the next time step.

The numerical grid system is shown in the Sketch 2 . In this procedure, the energy equation was solved first and having thus obtained the values for the dimensionless temperature at the end of the first time step, the temperature gradient was fed into the vorticity equation and this value of the vorticity at the nodes at the end of the first time step was fed to the stream function equation . The stream function equation was modified with the introduction of an artificial time dependence , since the ADI scheme is based on splitting the time step. The time step for the transient stream function scheme was chosen as 0.06, as the time step of this magnitude has been found by Stuart and Churchill [14] to be optimal for the stream function equation .

The stream function iterations were carried on until the maximum of the absolute difference in the values dropped below 5.e-04. When this value was attained, the solution was concluded to have reached an asymptotic steady state. This completed one iteration of

the equations, and the u and v velocities as well as the wall vorticities were calculated using the stream functions as explained later on. These values were used in the calculation of the next iterative cycle. Although the values of the u and v velocities lagged behind by a time step, the results were not affected as only the asymptotic steady state was of interest.

Further, the efforts to ensure that the u and v velocity computations were in line with the others as regards the time step would have entailed a large increase in the computer time , as some number of iterations had to be gone through , at every global iteration cycle, for the above purpose.

This sequence of iterations were repeated until the maximum absolute variation of the variables dropped below 0.1 percent.

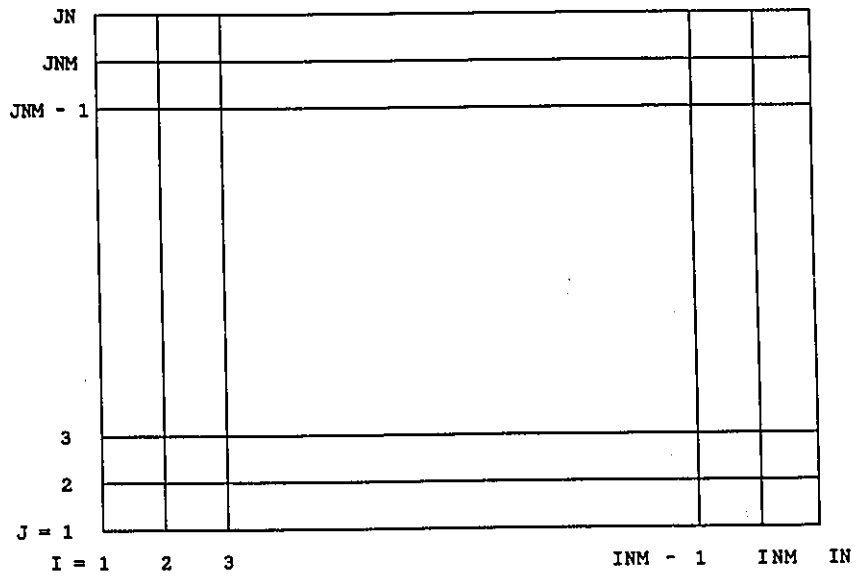
3.2 Boundary Conditions

3.2.1 Vorticity Boundary Conditions

The second order Taylor series approximations were used for the wall vorticity boundary conditions. As pointed out by Lin [10], the second order vorticity approximations required that the velocities be computed with second order accuracy.

This procedure is more accurate than the first order computations . The first order approximations were considered to be stable as compared to the second order vorticity computations according to Roache [12], but these higher order approximations do become stable if correspondingly higher order of computations for the u and v velocities are resorted to.

The procedure for determining these higher order expressions for vorticity is explained below.



Sketch 2 : The Grid System

In the following equations, the partial derivatives are shown by means of subscripts.

For example,

$\frac{\partial \psi}{\partial y}$ was written as ψ_y, and ψ_{yy} represents the second order partial derivative with respect to y.

Using the Taylor series approximations for the stream function at the nodes that are next to the wall ,

$$\psi_{i,2} = \psi_{i,1} + \psi_y\Big|_{i,1} \Delta y + \psi_{yy}\Big|_{i,1} \frac{(\Delta y)^2}{2\ !} \qquad (3.5)$$

$$\psi_{i,3} = \psi_{i,1} + \psi_y\Big|_{i,1} (2\Delta y) + \psi_{yy}\Big|_{i,1} \frac{(2\Delta y)^2}{2\ !} \qquad (3.6)$$

Hence,

$$8\psi_{i,2} - \psi_{i,3} = 2\,(\Delta y)^2\,\psi_{yy}\Big|_{i,1} \qquad (3.7)$$

$$\Rightarrow \qquad \omega_{i,1} = -\left[\frac{4\,\psi_{i,2} - 0.5\,\psi_{i,3}}{2}\right] \qquad (3.8)$$

Thus we arrive at the second order approximations for the wall vorticity. Since the expressions for the vorticities of the other walls are derived in a similar manner, only the final expressions are given below.

Thus,

$$\omega_{i,jn} = -\left[\frac{4\,\psi_{i,jn\,m} - 0.5\,\psi_{i,jn-2}}{(\Delta y)^2}\right] \qquad (3.9)$$

$$\omega_{1,j} = -\left[\frac{4\,\psi_{2,j} - 0.5\,\psi_{3,j}}{(\Delta x)^2}\right] \qquad (3.10)$$

And

$$\omega_{in,j} = -\left[\frac{4\,\psi_{inm,\,j} - 0.5\,\psi_{in-2,j}}{(\Delta x)^2}\right] \qquad (3.11)$$

3.3 Velocity Computations :

The expressions for the velocities at the interior nodes are obtained by considering the Taylor series expansions for the stream function values at two successive neighbouring nodes as follows

$$\psi_{i,j+1} = \psi_{i,j} + \psi_y\Big|_{i,j} \Delta y \qquad (3.12)$$

$$\psi_{i,j-1} = \psi_{i,j} - \psi_y\Big|_{i,j} \Delta y \qquad (3.13)$$

$$\psi_{i,j+2} = \psi_{i,j} + \psi_y\Big|_{i,j} (2\Delta y) \qquad (3.14)$$

$$\psi_{i,j-2} = \psi_{i,j} - \psi_y\Big|_{i,j} (2\Delta y) \qquad (3.15)$$

HENCE,

$$8\psi_{i,j+1} - 8\psi_{i,j-1} - \psi_{i,j+2} + \psi_{i,j-2} = 12\, u_{i,j}(\Delta y) \qquad (3.16)$$

$\Rightarrow$

$$u_{i,j} = \frac{8\psi_{i,j+1} - 8\psi_{i,j-1} - \psi_{i,j+2} + \psi_{i,j-2}}{12\,(\Delta y)} \qquad (3.17)$$

Similarly,

$$v_{i,j} = \frac{-8\psi_{i+1,j} + 8\psi_{i-1,j} + \psi_{i+2,j} - \psi_{i-2,j}}{12\,(\Delta y)} \qquad (3.18)$$

3.4 Next To Wall Velocity Computations :

For the cavity, the u and v velocities on all the four walls were set to zero in conformity with the no slip boundary

conditions; however, the velocity computations for the line of nodes adjacent to the wall had to be computed by a modified procedure. This is explained below.

$$\psi_{i,3} = \psi_{i,2} + \psi_y\Big|_{i,2} \Delta y \qquad (3.19)$$

$$\psi_{i,4} = \psi_{i,2} + \psi_y\Big|_{i,2} (2\Delta y) \qquad (3.20)$$

$$\psi_{i,1} = \psi_{i,2} - \psi_y\Big|_{i,2} \Delta y \qquad (3.21)$$

$$\psi_{i,2} = \psi_{i,2} \qquad (3.22)$$

$$\Rightarrow \quad u_{i,2} = \frac{6\psi_{i,3} - \psi_{i,4} - 2\psi_{i,1} - 3\psi_{i,2}}{6\,(\Delta y)} \qquad (3.33)$$

Similarly ,

$$u_{i,jnm} = -\left[\frac{6\psi_{i,jn-2} - \psi_{i,jn-3} - 2\psi_{i,jn} - 3\psi_{i,jnm}}{6\,(\Delta x)}\right] \qquad (3.34)$$

$$v_{2,j} = -\left[\frac{6\psi_{3,j} - \psi_{4,j} - 2\psi_{1,j} - 3\psi_{2,j}}{6\,(\Delta x)}\right] \qquad (3.35)$$

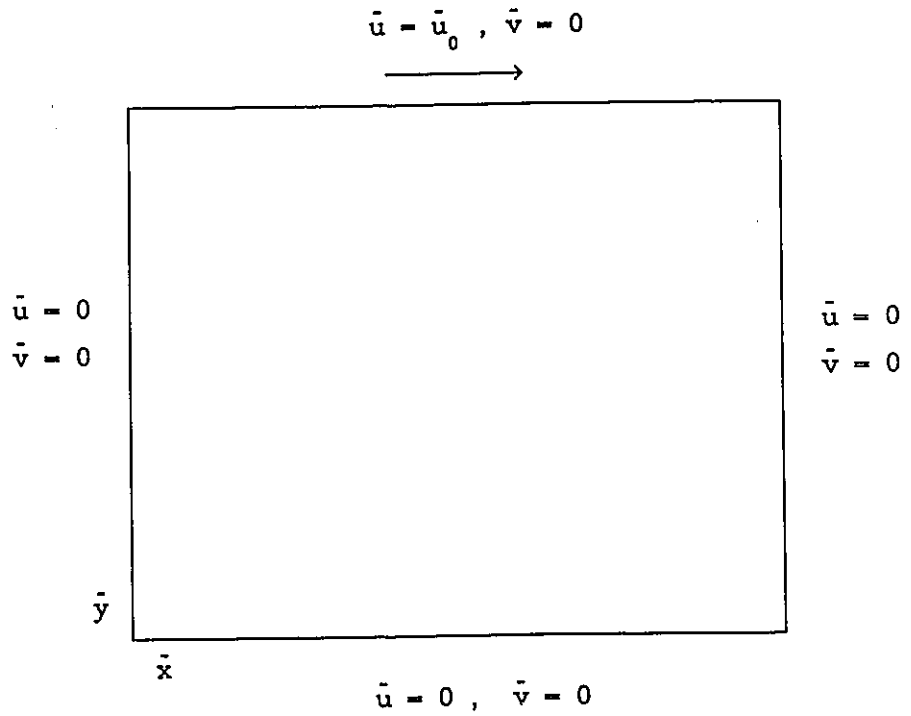
$$v_{inm,j} = \left[\frac{6\psi_{i,in-2} - \psi_{in-3,j} - 2\psi_{in,j} - 3\psi_{inm,j}}{6(\Delta x)} \right]$$

(3.36)

Before running the numerical experiments with the differentially heated square cavity, it was decided to test the validity of the false transient ADI scheme for the stream function equation. This was done by using the simple test case of a laminar driven cavity which is explained below

3.5 Laminar Driven Cavity :

Ed Lang (Ph.D.student, Univ. of Windsor) had used the Successive Over Relaxation (SOR) scheme for the stream function equation and ADI for the vorticity transport equation for the laminar driven cavity, which is shown Sketch 3. It was decided to compare the contours of the stream functions obtained by SOR scheme and the ADI scheme for the stream function equation for the case of the of the laminar driven cavity.



Sketch 3: The Laminar Driven Cavity

The SOR scheme is controlled by two unknowns namely the over-relaxation parameter as well as the time step . In ADI there was only one controlling parameter, namely the time step for the false transient equation; however, a time step of the order of 0.06 has been found by Stuart and Churchill [14] to be optimal for the above equation.

Further, second order wall vorticity boundary conditions were used as opposed to the first order boundary conditions in Lang's work as a higher order approximation gives a better accuracy. The non-dimensionalisation as well as the scheme and the boundary conditions used are described below.

The basic equations for the above problem are the continuity equation, and the x and y momentum equations as given by equations 2.1, 2.2 and 2.3.

Introducing the stream function and the vorticity equations as before and using the velocity of the plate $\bar{u}_0$ and the length $\bar{L}$ of the square cavity we get the following dimensionless parameters.

$$u = \frac{\bar{u}}{\bar{u}_0} \quad , \quad v = \frac{\bar{v}}{\bar{u}_0} \quad , \quad x = \frac{\bar{x}}{\bar{L}} \quad , \quad y = \frac{\bar{y}}{\bar{L}} \quad ,$$

$$\omega = \frac{\bar{\omega}}{\left[\frac{\bar{u}_0}{\bar{L}} \right]} \quad , \quad t = \frac{\bar{t}}{\left[\frac{\bar{L}}{\bar{u}_0} \right]} \quad , \quad Re = \frac{\bar{u}_0 \, \bar{L}}{\bar{\nu}}$$

(3.37)

Cross differentiating the momentum equations and introducing the definitions of stream function and vorticity to eliminate the pressure term,

$$\frac{D \omega}{D t} = \frac{1}{Re} \nabla^2 \omega \qquad (3.38)$$

The false transient stream function is the same as that for the differentially heated cavity which is equation 2.14.

For the equation 3.38 , according to the general form of the equation 2.15 , the coefficients are

$a = 0$, $b = 0$, $c = 1$ and $d = 0$.

The ADI development for the above equation is the same as

explained in section 3.1. The velocity computations were also carried out in the same manner as in sections 3.3.

The next to wall computations of the u and the v velocities were carried out in the same manner as before. the wall vorticities for all the walls except the one that moves with a velocity of 1.0 units (velocity u_0) were the same as in the cavity problem.

3.6 **Wall Vorticity for the Wall Driving the Flow** :

$$\omega_{i,jn} = \frac{-\left(8\,\psi_{i,jnm} - \psi_{i,jn-2} - 6\,u_0(\Delta y)\right)}{2\,(\Delta y)^2}$$

(3.39)

In the above expression, the value of u_0 is unity, since all the velocities are normalised with u_0, the plate velocity.

A 21 by 21 uniform grid and a 36 by 36 uniform grid were used for a Reynolds number of 10 . For a Reynolds number of 100, a 36 by 36 grid was used . The above grids were chosen so that the results could be compared with those of Ed Lang. The convergence criterion as before was set at 0.1 percent and a time step of 0.001 units was found to be sufficient in all of the above cases.

The contours of the stream function obtained by the ADI scheme and the SOR scheme were compared and they were in good agreement. For a Reynolds number of 10, two small cells rotating in a counter clockwise direction (Figures 21 to 23) near the lower corners of the cavity were observed, in addition to the main cell rotating in the clockwise direction, which is centrally placed. As the Reynolds number is increased to 100, the main clockwise rotating cell is

shifted towards the right top corner because of the greater effect of the inertial forces on the fluid in the cavity. This is due to the increased velocity of the plate driving the flow. Further, the counter clockwise cell on the lower right corner becomes greater in size because of the above discussed effect.

Having thus established the validity of the ADI scheme for the false transient stream function approach, the numerical experiments were carried out for the differentially heated square cavity for the cases documented by Lin and Nansteel [11] to test the competence of the mathematical and the subsequent numerical formulations and this is explained below.

3.7 Results and Discussion for the Square Cavity

The numerical experiments were performed based on the Rayleigh number, which is a ratio of the buoyancy forces to the viscous forces . A Rayleigh number of 1000 was chosen for our investigations so that the results could be compared to that of Lin and Nansteel [11].

The temperature of the hot wall was set by choosing the appropriate values of R. The experiments were run for R values of 0.4, 0.5, 0.55, 0.667 and 0. 75 with a 21 by 21 uniform grid, and the contours of the stream function agreed very well with the results of Lin and Nansteel [11] (Figures 1 to 14).

The numerical experiments were also run for a 41 by 41 grid for R values of 0.4 and 0.67 to test the grid dependency of the numerical model. The time step dependencies were also checked by lowering the time step by half and a comparison of the contours of the stream function for the lowered time step showed no perceptible

change.

The convergence criterion of 0.1 percent for all cases was not appropriate, and so the numerical experiments were run until successive orders of convergence yielded the same results. For example, for an R of 0.55 , the experiments were run for convergence orders of 0.1, 0.01 and 0.001 percent. The values of the stream function and the Nusselt numbers for the 0.01 percent convergence was concluded as being sufficient as these values did not vary by more than 4 percent as compared to those of 0.001 percent.

From the table 3.1 and contours (Figures 1 to 14), it is clear that the program is able to duplicate the results obtained by Lin and Nansteel [10]. The multicellular pattern for R of 0.4 , with the dominant cell on the hot wall side leads to a symmetric counter rotating cellular pattern at R = 0.5 and as R is increased to 0.55 , the cell becomes dominant on the cold wall side and as the R is increased to 0.75 the multicellular pattern leads to a unicellular structure as shown in the contours.

For the case of a value of 0.75 for R , there are indeed two small cells near the hot wall, similar to those with R value of 0.67, though these cells are smaller in size. This was not documented in Lin's thesis. The contours of dimensionless temperature did not show much variation with the grid refinement, but the two small cells of values 0.001 for the R value of 0.4 merge into one big cell for a 41 by 41 uniform grid.

3.8 Deviation from the Lin Algorithm

In the basic algorithm for the insulated horizontal walls, Lin had introduced a fictitious layer of cells beyond the

walls. This feature was used to write the finite difference equations for the zero temperature gradient in the horizontal walls. This additional step was found to be unnecessary as the same result could be obtained by using the Newmann boundary condition in the Thomas algorithm , which resulted in trimming the time needed for the algorithm development.

Further, it was found that Lin had written separate equations for the wall nodes and the next to wall nodes for the vorticity , stream function and the energy equations. These separate equations were found to be entirely unnecessary as either a Dirichlet or Newmann boundary condition specification for the walls would have automatically resulted in the evaluation of the values of these variables at the boundary and these modifications were incorporated in the computer program.

The aforementioned modifications resulted in the minimisation of the time necessary for the algorithm development and computer code incorporation in Fortran and later on was found to be a great time saver when the non-uniform grid algorithm was written.

R	Grid	$\psi_{max} * 10^3$	$\psi_{min} * 10^3$	Nu_h	Nu_c
0.4	21 * 21	1.5138	-26.774	1.008	1.008
	41 * 41	1.485	-26.797	1.008	1.007
Lin	21 * 21	1.5	-26.8		
0.5	21 * 21	10.71	-10.71	1.003	1.003
	25 * 25n	10.73	-10.73	1.019	1.019
	29 * 29n	10.79	-10.79	1.010	1.010
Lin	21 * 21	10.9	-10.9		
0.55	21 * 21	18.22	-4.799	1.005	1.007
Lin	21 * 21	18.4	-4.8		
0.67	21 * 21	38.47	-0.27	1.020	1.02
	41 * 41	38.53	-0.2695	1.019	1.02
	25 * 25n	38.77	-0.2643	1.016	1.02
Lin	21 * 21	38.6	-0.27	1.018	1.018
0.75	21 * 21	52.8	-4.028	1.038	1.042
Lin	21 * 21	53.2			

Table 3.1 Results for the Differentially Heated Cavity

Since the results from Lin [10] were checked with a similar grid structure, namely the uniform grid , a non uniform grid approach was also tried out to see if a better appreciation of the shear stresses could be obtained by bunching the nodes closer to the wall. Hence, the non uniform approach was attempted as explained below.

3.9 The Non - uniform Grid Algorithm

The need for a mesh system that could be used for flow situations which have large flow gradients such as a boundary layer flow resulted in the development of the non-uniform grid algorithm. The simplest method of incorporation was the use of a rectangular coordinate system with the flexibility to change the mesh spacings.

There are several restrictions for the above scheme as pointed by Roache [12] and they are the deterioration of the formal order of convergence, stability considerations, programming time and chances for error.

The formal order of truncation error deteriorates rapidly with large changes in the grid spacings. This means that the changes in the grid spacings have to be smooth. Fluent [4] recommends that the changes in grid spacings between adjacent grids be maintained between 70 and 130 percent in the entire computational domain.

The maximum time step is dependent on the lowest grid spacing near the wall [12] and this imposes a severe restriction on the smallest grid spacing that could be chosen for the investigations.

The mathematical treatment for the non-uniform algorithm was similar to that of the uniform grid.

The finite difference form of the equation 2.15 for the non-uniform grid scheme is given below. As before, the forward differencing has been applied for the time derivatives and central differencing, for the space derivatives. Thus,

For the first half time step,

$$\frac{\xi_{i,j}^{n+1/2} - \xi_{i,j}^{n}}{(\Delta t/2)} =$$

$$- a_{i,j}\left[\frac{\xi_{i+1,j}^{n+1/2}(x_i - x_{i-1})^2 - \xi_{i-1,j}^{n+1/2}(x_{i+1} - x_i)^2}{(x_{i+1} - x_i)(x_i - x_{i-1})(x_{i+1} - x_{i-1})} \right.$$

$$\left. - \frac{\xi_{i,j}^{n+1/2}\left[(x_i - x_{i-1})^2 - (x_{i+1} - x_i)^2 \right]}{(x_{i+1} - x_i)(x_i - x_{i-1})(x_{i+1} - x_{i-1})} \right]$$

$$- b \frac{\partial \xi_{i,j}^{n}}{\partial y} + c \frac{\partial^2 \xi_{i,j}^{n}}{(\Delta y)^2}$$

$$+ 2\,c \left[\frac{\xi^{n+1/2}_{i+1,j}(x_i - x_{i-1}) - \xi^{n+1/2}_{i-1,j}(x_{i+1} - x_i)}{(x_{i+1} - x_i)(x_i - x_{i-1})(x_{i+1} - x_{i-1})} - \xi^{n+1/2}_{i,j} \left[\frac{(x_{i+1} - x_{i-1})}{(x_{i+1} - x_i)(x_i - x_{i-1})(x_{i+1} - x_{i-1})} \right] \right]$$

$$+ d$$

(3.40)

Rearranging the above equations , to express the above equation in terms of the coefficients at the nodes i, i+1 and i-1, we get,

$$\left[\frac{a_{i,j}\ \Delta t\ (x_i - x_{i-1})^2}{2\ (x_{i+1} - x_i)(x_i - x_{i-1})(x_{i+1} - x_{i-1})} - \frac{c\,\Delta t\,(x_i - x_{i-1})}{(x_{i+1} - x_i)(x_i - x_{i-1})(x_{i+1} - x_{i-1})} \right] \xi^{n+1/2}_{i+1,j}$$

$$+ \left[1 - \frac{a_{i,j}\ \Delta t \left[(x_i - x_{i-1})^2 - (x_{i+1} - x_i)^2 \right]}{2\,(x_{i+1} - x_i)(x_i - x_{i-1})(x_{i+1} - x_{i-1})} \right]$$

$$+ \left[\frac{c\,\Delta t\,(x_{i+1} - x_{i-1})}{(x_{i+1} - x_i)(x_i - x_{i-1})(x_{i+1} - x_{i-1})} \right] \xi_{i,j}^{n+1/2}$$

$$+ \left[- \frac{a_{i,j}\,\Delta t\,(x_{i+1} - x_i)^2}{2\,(x_{i+1} - x_i)(x_i - x_{i-1})(x_{i+1} - x_{i-1})} \right.$$

$$\left. - \frac{c\,\Delta t\,(x_{i+1} - x_i)}{(x_{i+1} - x_i)(x_i - x_{i-1})(x_{i+1} - x_{i-1})} \right] \xi_{i-1,j}^{n+1/2}$$

$$= - \frac{\Delta t}{2} \left[\frac{\partial^2 \xi_{i,j}^n}{\partial y^2} - b_{i,j} \frac{\partial \xi_{i,j}^n}{\partial y} \right]$$

$$+ \frac{d\,\Delta t}{2} + \xi_{i,j}^n$$

(3.41)

The above equation can be fed directly into the Thomas algorithm and the values of the variable at every node in the domain after the first half time is obtained , with the x directional dependencies being taken into account.

Similarly, for the second half time step, taking the y directional variations at the new time step into account,

$$\frac{\xi_{i,j}^{n+1} - \xi_{i,j}^{n+1/2}}{(\Delta t / 2)} =$$

$$- a_{i,j} \left[\frac{\xi_{i,j+1}^{n+1} (y_j - y_{j-1})^2 - \xi_{i,j-1}^{n+1} (y_{j+1} - y_j)^2}{(y_{j+1} - y_j)(y_j - y_{j-1})(y_{j+1} - y_{j-1})} \right.$$

$$\left. - \frac{\xi_{i,j}^{n+1} \left[(y_j - y_{j-1})^2 - (y_{j+1} - y_j)^2 \right]}{(y_{j+1} - y_j)(y_j - y_{j-1})(y_{j+1} - y_{j-1})} \right]$$

$$- b_{i,j} \frac{\partial \xi_{i,j}^{n+1/2}}{\partial x} + c \frac{\partial^2 \xi_{i,j}^{n+1/2}}{(\Delta x)^2}$$

$$+ 2c \left[\frac{\xi_{i,j+1}^{n+1} (y_j - y_{j-1}) - \xi_{i,j-1}^{n+1} (y_{j+1} - y_j)}{(y_{j+1} - y_j)(y_j - y_{j-1})(y_{j+1} - y_{j-1})} \right.$$

$$\left. - \frac{\xi_{i,j}^{n+1} \left[(y_{j+1} - y_{j-1}) \right]}{(y_{j+1} - y_j)(y_j - y_{j-1})(y_{j+1} - y_{j-1})} \right]$$

$$+ \; d$$

(3.42)

Rearranging the above equations , to express the above equation in terms of the coefficients for values of the variable for the new time at the nodes j+1, j and j-1 we get

$$\left[\frac{a_{i,j} \; \Delta t \; (y_j - y_{j-1})^2}{2 \; (y_{j+1} - y_j) \; (y_j - y_{j-1}) \; (y_{j+1} - y_{j-1})} \right.$$

$$\left. - \; \frac{c \; \Delta t \; (y_i - y_{i-1})}{(y_{i+1} - y_i) \; (y_i - y_{i-1}) \; (y_{i+1} - y_{i-1})} \right] \; \xi_{i,j+1}^{n+1}$$

$$+ \left[1 - \frac{a_{i,j}\ \Delta t \left[\left(y_j - y_{j-1} \right)^2 - \left(y_{j+1} - y_j \right)^2 \right]}{2 \left(y_{j+1} - y_j \right) \left(y_j - y_{j-1} \right) \left(y_{j+1} - y_{j-1} \right)} \right.$$

$$\left. + \frac{c\ \Delta t \left(y_{j+1} - y_{j-1} \right)}{\left(y_{j+1} - y_j \right) \left(y_j - y_{j-1} \right) \left(y_{j+1} - y_{j-1} \right)} \right] \xi_{i,j}^{n+1}$$

$$+ \left[- \frac{a_{i,j}\ \Delta t \left(y_{j+1} - y_j \right)^2}{2 \left(y_{j+1} - y_j \right) \left(y_j - y_{j-1} \right) \left(y_{j+1} - y_{j-1} \right)} \right.$$

$$\left. - \frac{c\ \Delta t \left(y_{j+1} - y_j \right)}{\left(y_{j+1} - y_j \right) \left(y_j - y_{j-1} \right) \left(y_{j+1} - y_{j-1} \right)} \right] \xi_{i,j-1}^{n+1}$$

$$= - \frac{\Delta t}{2} \left[\frac{\partial^2 \xi_{i,j}^{n+1/2}}{\partial x^2} - b_{i,j} \frac{\partial \xi_{i,j}^{n+1/2}}{\partial x} \right]$$

$$+ \frac{d\ \Delta t}{2} + \xi_{i,j}^{n+1/2}$$

(3.43)

Thus, the values of the variable at the new time step taking both the x and y directional dependencies into account are obtained and this concludes the algorithm development for the non-uniform grid.

3.10 Boundary Conditions

The boundary conditions for the wall vorticities for the non-uniform grid are calculated by using Taylor series approximations as explained for the uniform grid, with the non-uniform grid spacings being taken into account.

Thus,

$$\omega_w = \frac{-(8\,\psi_{w+1} - \psi_{w+2})}{\left[3.5(y_{w+1} - y_w)^2 - (y_{w+1} - y_w)(y_{w+2} - y_{w+1}) - \frac{(y_{w+2} - y_{w+1})^2}{2} \right]} \qquad (3.44)$$

In the above equation ,the subscript w indicates the wall. The y in the denominator indicates that the above expression is for the horizontal walls. Changing y to x in the above expression gives the wall vorticity for the vertical walls.

The Taylor series approximations for the non-uniform grid are used to arrive at the expressions for the velocities and the procedure is the same as for an uniform grid. Thus,

$$u_{i,j} = \frac{(8\,\psi_{i,j+1} - 8\,\psi_{i,j-1} - \psi_{i,j+2} + \psi_{i,j-2})}{(8y_{j+1} - 8y_{j-1} - y_{j+2} + y_{j-2})} \qquad (3.45)$$

$$v_{i,j} = \frac{-\left(8\,\psi_{i+1,j} - 8\,\psi_{i-1,j} - \psi_{i+2,j} + \psi_{i-2,j} \right)}{\left(8x_{i+1} - 8x_{i-1} - x_{i+2} + x_{i-2} \right)} \qquad (3.46)$$

The next to wall velocities are

$$u_{i,2} = \frac{\left(6\,\psi_{i,3} - \psi_{i,4} - 3\psi_{i,2} \right)}{\left[5\,(y_3 - y_2) - (y_4 - y_3) + 2\,(y_2 - y_1) \right]} \qquad (3.47)$$

$$u_{i,jnm} = \frac{-\left(6\,\psi_{i,jn-2} - \psi_{i,jn-3} - 3\psi_{i,jnm} \right)}{\left[5(y_{jnm} - y_{jn-2}) - (y_{jn-2} - y_{jn-3}) + 2\,(y_{jn} - y_{jnm}) \right]} \qquad (3.48)$$

$$v_{i,2} = \frac{-\left(6\,\psi_{3,j} - \psi_{4,j} - 3\psi_{2,j} \right)}{\left[5\,(x_3 - x_2) - (x_4 - x_3) + 2\,(x_2 - x_1) \right]} \qquad (3.49)$$

$$v_{i,jnm} = \frac{\left(6\,\psi_{in-2,j} - \psi_{in-3,j} - 3\psi_{inm,j} \right)}{\left[5(x_{in-2} - x_{inm}) - (x_{in-3} - x_{in-2}) + 2\,(x_{inm} - x_{in}) \right]} \qquad (3.50)$$

The ADI scheme that was developed for the non-uniform grid arrangement was run with a uniform grid spacing. The contours of the stream function for the above case for an R of 0.5 were identical

to those for the uniform grid algorithm with the same grid spacing.

The Nusselt numbers for the hot and cold walls as well as the maximum and minimum values of the stream function were within 1 percent with those of the uniform grid arrangement in the uniform grid scheme and this provided a check for the validity of the non - uniform algorithm.

A 25 by 25 non - uniform grid was used for R values of 0.5 and 0.67 and a 29 by 29 non - uniform grid was used for an R of 0.5. These non - uniformities were introduced near both walls , for a length of 0.15 in both directions to get a better appreciation of the shear stresses (Figures 15 to 20). The grid spacing at the nodes nearest to the wall was 40 percent of that of the uniform grid scheme for the 25 by 25 non-uniform grid scheme and 20 percent for the 29 by 29 non-uniform grid scheme. The maximum and minimum stream function values as well as the Nusselt number values were compared with those of a 21 by 21 uniform grid for the corresponding cases , and these are given in table 3.2 .

R	Grid	ψ_{max}^{z}	ψ_{min}^{z}	Nu_h^{z}	Nu_c^{z}
0.5	25 * 25n	0.186	0.186	1.59	1.59
	29 * 29n	0.741	0.741	0.64	0.64
0.67	25 * 25n	0.954	3.5	1.2	0.05

Table 3.2 Comparison of the Non-uniform Results With That of The Uniform Scheme

As before, the subscript n indicates a non uniform grid.

The contours of the stream function and temperature (Figures 15 to 20) show no perceptible changes in the temperature profiles. However , for the cases of R values of 0.5 and 0.67 the innermost cell is a wider than the corresponding cell for the 21 by 21 uniform grid case.

This concludes the investigation of the differentially heated square cavity and the following chapter gives the use of the program developed for the above case for the flow of water over an ice sheet .

CHAPTER 4

FLOW OVER AN ICE SHEET : PROGRAM DEVELOPMENT AND VALIDATION

4.1 Introduction

The flow of cold water over an horizontal ice sheet has been investigated earlier by Wilson and Sarma [18]. In their investigations the flow of water over an infinitely long ice sheet was analysed. The infinitely long ice sheet was numerically modelled by using extrapolative boundary conditions at the downstream boundary. At large free stream velocities, they obtained solutions similar to that of forced convection over a flat plate. At lower free stream velocities, recirculation cells were observed and the numerical solution was oscillatory. As mentioned on page 7, it was not clear whether the recirculation cells were a flow phenomenon or a numerical instability. Hence, the ADI scheme , the validity of which was established by its ability to reproduce the results of Lin and Nansteel and to which some modifications and the subsequent capability of non-uniformity was incorporated, was used for the above investigation.

4.2 Mathematical Formulation

The basic equations for the flow over an ice sheet are the same as those of the differentially heated square cavity, namely the continuity equation, the energy equation and the x and y momentum equations and the basic assumptions are the same as those for the rectangular cavity.

The normalisation technique for the velocities and the lengths were the same as that for the laminar driven cavity namely ,

$$u = \frac{\bar{u}}{\bar{u}_0} \quad , \quad v = \frac{\bar{v}}{\bar{u}_0} \quad , \quad x = \frac{\bar{x}}{\bar{L}} \quad , \quad y = \frac{\bar{y}}{\bar{L}} \quad ,$$

$$\omega = \frac{\bar{\omega}}{\left[\frac{\bar{u}_0}{\bar{L}} \right]} \quad , \quad t = \frac{\bar{t}}{\left[\frac{\bar{L}}{\bar{u}_0} \right]} \quad , \quad Re = \frac{\bar{u}_0 \, \bar{L}}{\bar{\nu}}$$

(4.1)

In the above expressions, $\bar{u}_o$ is the free stream velocity, $\bar{L}$ is the length of the domain in the $\bar{x}$ direction, which has been choosen as the characteristic length, so that the ratio $\frac{\bar{L}}{\bar{u}_0}$, the time taken by the fluid particle to roll over the domain could be chosen as the characteristic time.

However, the temperature normalisations and definition of the Rayleigh number were done in the same manner as that of flow in rectangular cavity and these are shown below

$$\phi = \frac{\bar{T} - \bar{T}_c}{\bar{T}_h - \bar{T}_c}$$

$$Ra = \frac{g \, \alpha_1 \rho_m L^3 \, (\bar{T}_h - \bar{T}_c)^q}{\rho_c \, \nu \, \alpha}$$

$$Pr = \frac{\nu}{\alpha} \qquad R = \frac{\bar{T}_m - \bar{T}_c}{\bar{T}_h - \bar{T}_c}$$

(4.2)

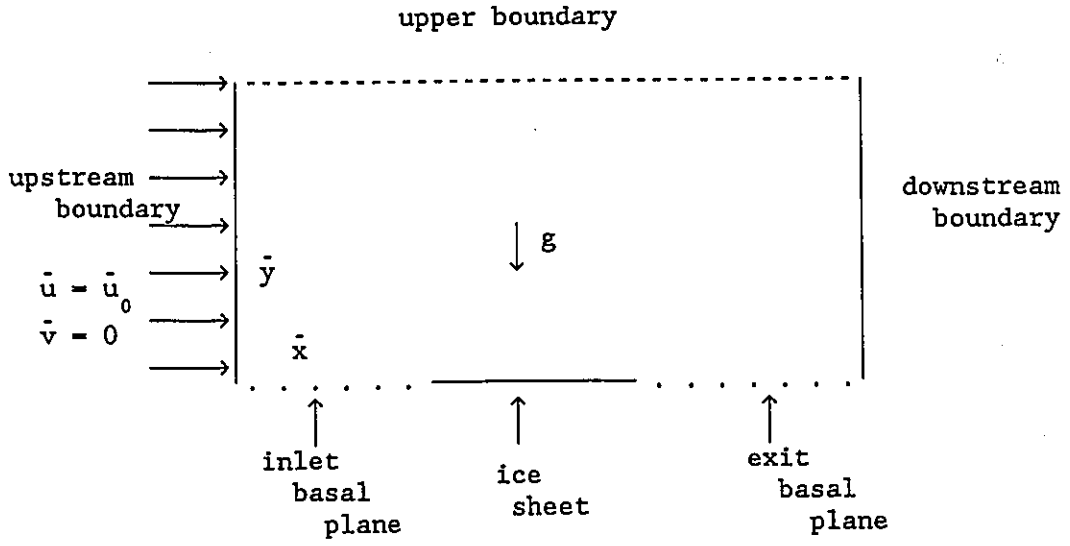
The nomenclature for the above expressions are the same as that for the ice sheet. The cold temperature is that of the ice sheet , namely 0 ^{0}C , and the hot temperature is that of the free stream temperature of the water.

Normalising the equations using the above parameters and eliminating the pressure term , the following equations are obtained. Equation 4.3 represents the vorticity equation, 4.4 , the energy equation and 4.5 , the stream function equation.

$$\frac{D\,\omega}{D\,t} = \frac{Ra}{Pr\ Re^2}\ q\,|\,\phi - R\,|^{q-2}(\phi - R)\frac{\partial \phi}{\partial x} + \frac{1}{Re}\nabla^2\omega \qquad (4.3)$$

$$\frac{D\,\phi}{D\,t} = \frac{1}{Pr\ Re}\nabla^2\phi \qquad (4.4)$$

$$\nabla^2\psi = -\omega \qquad (4.5)$$



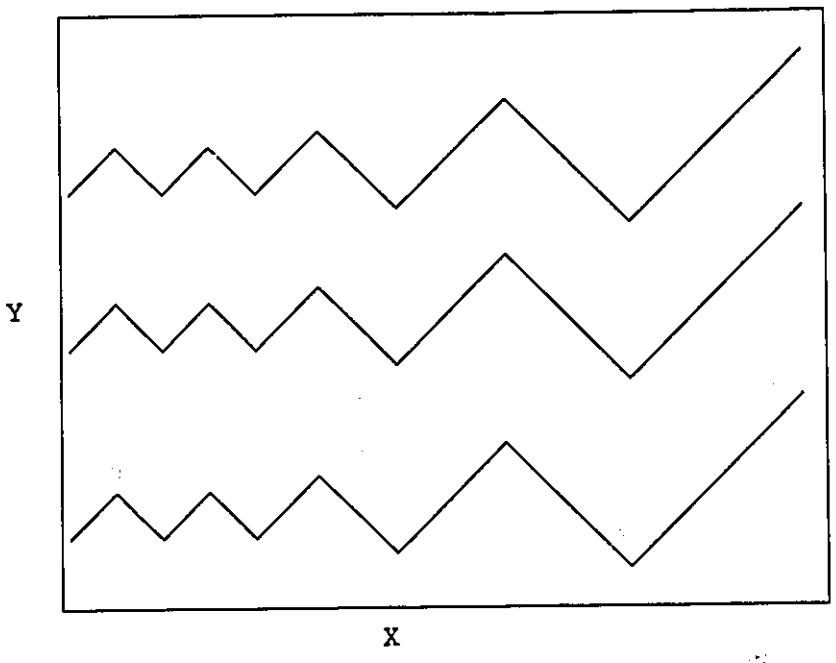
Sketch 4 : Flow Over an Ice Sheet

4.3 Numerical Formulation

The computational domain is shown in Sketch 4. The ADI scheme as described in section 3.7 was used for the governing equations , with the stream function equation being solved using the false transient algorithm , as explained in the differentially heated square cavity. The energy and the vorticity equations were solved based on the upwind difference scheme and the program was unchanged except for the implementation of the boundary conditions. The reason for using the upwinding difference scheme was the occurence of wiggles, which is explained below.

4.4 Wiggles

When the appropriate boundary conditions were set forth for the flow over an ice sheet, spatial oscillations in the flow situation called wiggles were encountered as illustrated in Sketch 5. Wiggles are associated with the post-shock oscillations of methods using centered space derivatives [12]. Wiggles were also associated with the long term incompressible flow calculations. Wiggles have been associated with nonlinearities and also linear instabilities in the transient calculations.



Sketch 5 : Contours of Stream Function Illustrating Wiggles

There are several procedures that have been utilised earlier to overcome this problem of wiggles. Three methodologies were attempted for this case to eliminate wiggles. They were Newmann boundary conditions, Shapiro and O' Brien formulation [12] and the Upwinding scheme. The Newmann boundary condition refers to the application of gradients on the boundary instead of setting the values. This method did not eliminate the wiggles and so the Shapiro and O'Brien condition [12], which basically neglects diffusion over the last two node columns and is basically a technique for inviscid flows, which is recommended for high Reynolds number flows, was tried out. This did not eliminate wiggles and the upwinding difference scheme had to be resorted to which successfully eliminated wiggles.

The upwind differencing scheme was used for the energy equation as well as the vorticity equation because using upwinding in

one of these equations did not eliminate wiggles.

For example, in the equation below, a is a coefficient and ξ is the variable for which upwinding has to be applied.

If a is positive,

$$- a \frac{\partial \xi}{\partial x} = - a \frac{\xi_{i+1,j} - \xi_{i,j}}{(x_{i+1} - x_i)} \qquad (4.21)$$

if a is negative, we get

$$- a \frac{\partial \xi}{\partial x} = - a \frac{\xi_{i,j} - \xi_{i-1,j}}{(x_i - x_{i-1})} \qquad (4.22)$$

Thus, the upwinding scheme takes into account the variation of the flow parameters from the upstream end of the flow for spatial derivatives .

Incorporation of the above evaluated parameters resulted in the successful elimination of wiggles.

4.5 Boundary Conditions

4.5.1 Introduction

The free stream velocity was used to determine the inlet stream function. The vorticity at the inlet was set as zero , because the leading edge was at some distance downstream in the computational domain and the free stream conditions were assumed. The inlet temperature was set as the free stream temperature.

The inlet and exit basal planes had symmetry boundary conditions applied to them. The ice sheet region had the boundary conditions which were determined as follows. The melt velocity was

used to determine the stream function, which in turn was used to determine the vorticities. The temperature was set to that of the ice sheet. The wall velocities were determined by equating the rate of heat conducted into the ice to the rate of the melting of the ice sheet.

Extrapolative boundary conditions were used for the downstream boundary. The free stream velocity was again used to set the stream function at the upper boundary and the vorticity was set as zero .

The exact equations that were used to describe the boundary will be discussed in the following section.

The mathematical and numerical boundary conditions are discussed together. The boundary conditions for the flow over ice sheet differ considerably from the case of flow in a differentially heated cavity and this is described below.

4.5.2 Upstream Boundary

Since the velocities are normalised by the free stream velocity at the inlet, the U velocity at the inlet plane is always unity .

$$U = 1.0$$

$$V = 0$$

This information is used to determine the values of the stream function at the inlet. Numerically, this is expressed as

$$\psi_{1,j+1} = \psi_{1,j} + 1.0 \, (y_{j+1} - y_j)$$

where the reference value of the stream function is set to zero. Thus,

$$\psi_{1,1} = 0$$

The vorticity is set to zero and the dimensionless temperature is set to that of the free stream. Numerically,

$$\omega_{1,j} = 0 \qquad (4.6)$$

$$\phi_{1,j} = 1.0$$

4.5.3 Inlet Basal Plane

A symmetric leading edge is assumed and hence the gradient of the u velocity in the y direction is set as zero. No flow is allowed to cross the boundary and hence the v velocity is set to zero.

$$\frac{\partial u}{\partial y} = 0$$

$$v = 0$$

Numerically $\psi_{i,1} = 0.$(for all i's corresponding to the upstream region)

Also, the vorticity is set to zero and the gradient of temperature in the y direction is set to zero and hence

$$\omega = 0 \qquad (4.7)$$

$$\frac{\partial \phi}{\partial y} = 0$$

4.5.4 Exit Basal Plane

A symmetric trailing edge is assumed as for the inlet basal plane and the boundary conditions are as follows :

The gradient of temperature in the y direction is set to zero .

$$\frac{\partial \phi}{\partial y} = 0$$

The boundary condition for the stream function expressed numerically is as follows

$$\psi_{i,1} = \psi_{k_e,1}$$

where k_e is the node corresponding to the end of the ice sheet in the main flow direction.

As before, the vorticity is set as zero due to the symmetric boundary condition

$$\omega = 0 \qquad (4.8)$$

4.5.5 Downstream Boundary

Linear extrapolation was used for the downstream boundary for temperature, stream function and vorticity.

4.5.6 Upper Boundary

The temperature of the upper plane was set as the free stream temperature and the u velocity was set as that of free stream velocity. Hence,

$$\phi_{i,jn} = 1.0 \quad \text{and} \qquad (4.9)$$

$$\psi_{i,jn} = \psi_{i,jnm} + 1.0\,(y_{jn} - y_{jnm})$$

The vorticity is set to zero and

$$w = 0$$

4.5.7 Ice Sheet

The wall velocity was based on equating the heat conducted to the ice sheet as being required to melt the ice. Hence,

$$\bar{v}_w = \frac{K}{\rho_c L_f} \frac{\partial \bar{T}}{\partial \bar{y}} \qquad (4.10)$$

$\bar{v}_w$ is the wall velocity, K is the thermal conductivity of water at 0^0 C and L_f, the latent heat of fusion of ice. Introducing the non dimensional parameters, we get

$$v_w = \frac{(\bar{T}_h - \bar{T}_c)\,K}{\nu\,\rho_c\,L_f\,Re} \frac{\partial \phi}{\partial y} \qquad (4.11)$$

The boundary conditions are

$$\psi_{m+1,1} = \psi_{m,1} - \int_{x_m}^{x_{m+1}} v_w \, dx$$

$$k_i < m < k_e \qquad (4.12)$$

Where k_i is the node in the x direction corresponding to the start of the ice sheet in the stream wise sense. The Trapezoidal rule was used for the above integration of the wall velocity.

The dimensionless temperature is set as zero and thus

$$\phi_{m,1} = 0 .$$

The vorticity boundary conditions were the same as that for the differentially heated square cavity with the additional expression for the second derivative of the stream function in the x direction. The u and v velocities were calculated in the same manner as that for the rectangular cavity.

Hence,

$$\omega_w = \frac{-(8\psi_{w+1} - \psi_{w+2} - 7\psi_w)}{\left[3.5(y_{w+1} - y_w)^2 - (y_{w+1} - y_w)(y_{w+2} - y_{w+1}) - \frac{(y_{w+2} - y_{w+1})^2}{2}\right]}$$

$$- \frac{2\{\psi_{i+1}(x_i - x_{i-1}) + \psi_{i-1}(x_{i+1} - x_i) - \psi_i(x_{i+1} - x_{i-1})}{(x_i - x_{i-1})(x_{i+1} - x_i)(x_{i+1} - x_{i-1})}$$

(4.13)

The velocities were calculated as for the rectangular cavity by using Taylor series expansions . Thus,

$$u_{i,j} = \frac{(8\psi_{i,j+1} - 8\psi_{i,j-1} - \psi_{i,j+2} + \psi_{i,j-2})}{(8y_{j+1} - 8y_{j-1} - y_{j+2} + y_{j-2})}$$

(4.14)

$$v_{i,j} = \frac{- (8\psi_{i+1,j} - 8\psi_{i-1,j} - \psi_{i+2,j} + \psi_{i-2,j})}{(8x_{i+1} - 8x_{i-1} - x_{i+2} + x_{i-2})}$$

(4.15)

The next to wall velocities were

$$u_{i,2} = \frac{(6\psi_{i,3} - \psi_{i,4} - 3\psi_{i,2} - 2\psi_{i,1})}{\left[5 (y_3 - y_2) - (y_4 - y_3) + 2 (y_2 - y_1) \right]}$$

(4.16)

$$u_{i,jnm} = \frac{- (6\psi_{i,jn-2} - \psi_{i,jn-3} - 3\psi_{i,jnm} - 2\psi_{i,jn})}{\left[5(y_{jnm} - y_{jn-2}) - (y_{jn-2} - y_{jn-3}) + 2 (y_{jn} - y_{jnm}) \right]}$$

(4.17)

$$v_{i,2} = \frac{- (6\psi_{3,j} - \psi_{4,j} - 3\psi_{2,j} - 2\psi_{1,j})}{\left[5 (x_3 - x_2) - (x_4 - x_3) + 2 (x_2 - x_1) \right]}$$

(4.18)

$$v_{i,jnm} = \frac{(6\psi_{in-2,j} - \psi_{in-3,j} - 3\psi_{inm,j} - 2\psi_{in,j})}{\left[5(x_{in-2} - x_{inm}) - (x_{in-3} - x_{in-2}) + 2 (x_{inm} - x_{in}) \right]}$$

(4.19)

The temperature gradient for the calculation of the wall velocity was calculated using the Taylor series expansions again ana

] thus

$$\left.\frac{\partial \phi}{\partial y}\right|_{i,1} = \frac{(8 \phi_{i,2} - \phi_{i,3} - 7 \phi_{i,1})}{(7 (y_2 - y_1) - (y_3 - y_2))}$$

(4.20)

This concludes the section on the boundary conditions and the computer code was then tested against that of the Blasius and Pohlhausen profiles and this is explained in the following section.

4.6 Validation by Comparison to Blasius and Pohlhausen Profiles

The above formulation was validated as follows. The melt velocity was set as zero along the ice sheet and the corresponding stream function and vorticity equations were generated to simulate the flow of cold water over an isothermal surface at 0^0 C. The velocity profiles were compared with those predicted by Blasius and the temperature profiles were compared with those of Pohlhausen for a Prandtl number of 13.0 (Schlichting) [15] .

The velocity profile comparison with the Blasius profile shows that the computed boundary layer thickness is greater near the leading edge (Figures 24 to 31) and the profile approaches that of Blasius near the trailing edge. This effect is due to several approximations made in the Blasius approximation [15]. The v velocity at the inlet plane of the ice sheet is indeterminate , whereas the computed solution lets the v velocity develop as part of the solution. This and other approximations result in the Blasius profile being inaccurate near the leading edge and it gets closer further downstream as explained by Kuo [8] and Carrier et.al.,[3] and thus the computed

solution was found to be valid.

4.7 Increased U Velocity

Furthermore, it was observed that the computed u velocity was higher than the inlet u velocity , outside the boundary layer. For example , at the location of x of 0.0814 , there was an increase of 0.6 percent. Though this increase is very small, and has not been documented before, it was concluded from the nature of the stream lines that the flow was accelerating as it reaches the ice sheet, and this behaviour contributes to the increased u velocity. This conclusion was reached by recognizing that the v velocity components do undergo an increase in magnitude because the stream lines that describe the flow situation are deflected upwards as they approach the leading edge of the plate. It is postulated that there is an increase in the velocity in the x direction as well.

The Pohlhausen comparison was similar to that of Blasius in the sense of better correlation between the computed and the theoretically predicted temperature profiles downstream of the flat plate.

Thus, the numerical code was validated by comparing the results with an earlier established case and the ice sheet was modelled as explained earlier and the next chapter is based on the results of the flow over an ice sheet.

CHAPTER 5

RESULTS, DISCUSSION AND NUMERICAL INSTABILITIES

5.1 Introduction

The following chapter gives a detailed account of the results of the numerical modelling of the flow of cold water above an ice sheet. The associated numerical instabilities and the subsequent discussions are also documented in detail. The most difficult and time consuming portions were the attempts to choose the appropriate boundary conditions. Melting walls, free flight top surface planes, regions before the leading and the trailing edges had to be modelled. Free flight is a terminology used to describe the modelling of the top surface of the fluid in the numerical simulation of the wind tunnel experiments. Some very unique flow instabilities called as the wiggles [12], had to be overcome as well. The initial attempts to model an infinitely long ice sheet resulted in numerical instabilities as well. The results were influenced by the length of the ice sheet and the grid spacing (Figures 31 to 42) and these are discussed in detail below.

5.2 Flow situation at high free stream velocities

The final steady state solution is attained due to the balance between the inertial forces, the viscous forces and the buoyancy forces. For the higher free stream velocities, the inertial forces outweigh the buoyancy forces and the viscous forces and this results in a flow situation similar to the forced convection case.

5.3 False Recirculation

For the runs with the lower free stream velocities, a recirculation cell was seen near the leading edge, when a relatively

coarse mesh in the y direction was used . This cell disappeared when a finer mesh was used, which basically underlines the importance of grid dependencies in the numerical calculations.

5.4 Upper Boundary Condition

When a free flight boundary condition as described by Roache and Mueller [13] was used to model the upper boundary, the stream function was determined based on the free stream velocity at the inlet . Hence, the stream line is a constant on the upper surface and no flow was allowed to cross the boundary. This scheme proved to be disastrous in the lower free stream velocity runs, when the buoyancy terms become significant. This is because the flow lifts up, only to be pushed firmly down by the fixed stream line on the top surface . This resulted in a converged solution which was physically impossible for the nature of the flow situation that was simulated. Hence the boundary conditions were set to allow the flow to cross the boundary and to rejoin , if necessary, after travelling some distance along the domain.

5.5 Length of the Ice Sheet

The residuals for the vorticity , stream function and temperature were oscillatory as the grid was made finer for the numerical approximation of an infinitely long ice sheet. The infinitely long ice sheet, as discussed earlier was modelled by assuming extrapolative boundary conditions downstream. To ensure that the above phenomenon was a physical nature of the flow and not a numerical instability , a finite length of the ice sheet was attempted and no such oscillatory behaviour was observed. The length of the ice sheet was increased and there still was not any oscillatory behaviour

and so it was concluded that the numerical instabilities caused by the extrapolative boundary conditions were the reason for the oscillations in the residuals.

5.6 Grid Spacing

The grids in the x direction were made finer until the solutions looked physically relevant in the sense that the leading and trailing edge transitions on the stream lines were smooth. Though the melt velocity stream lines near the trailing edge change sharply, attempts to make the grid finer in this region alone results in the overall increase of the grid in the entire domain of computation to ensure that there are no abrupt changes in the grid spacings. This would mean a massive increase in the computer time , and all this would do is to make the stream lines a bit more smoother and it was decided to forego this grid refinement.

The grid spacing for the y grid nearest to the ice sheet was determined as described in Fluent [4]. This grid spacing was recommended by Schlichting to ensure the accurate computation of the shear stress on the wall. Since the energy equation was also involved in the computation , the minimal grid spacing was solved and the basic flow pattern was unchanged as indicated by the stream function profiles, but the minimum stream function value dropped by about ten percent. The grid spacing was again halved and the change in the minimum stream function was only 3.9 percent and since the stream function patterns were the same, the solution was concluded as grid spacing independent at the ice wall. The grid dependencies in the x and y direction were checked by increasing the grids in either direction and comparing the flow patterns each time making sure that

grid spacing variations were between 80 and 120 percent as recommended by Fluent [4], supported by Roache who has shown that the order of truncation decays with drastic variation. With every finer grid system, Schlichting's shear stress criterion was also met to ensure accurate shear stress computations.

The grid refinement was continued until two successive grid systems gave the same solutions, in that the contours of the stream functions as well as the values of the maximum and minimum stream functions, were checked. Convergence was said to have been reached once the values of the variables dropped below 1.e-03 . The convergence was earlier checked by going for a higher order of convergence and comparing the solutions and also leaving a few grids near the wall that are pretty close to the wall out of the residual calculation. It was clear that the residuals did drop down near the walls , but after many number of iterations, and the solution was not altered in any manner and ignoring a few grids near the wall resulted in saving of computer time. The computer time was also trimmed by using linear interpolations and extrapolations of a similar case as a starting solution. Typically, for the satisfactory convergence of the residuals, about 25,000 iterations had to be gone through, with the time step being 1.e-04 for a free stream velocity of 0.1 m/s and an R of 0.2 , if the numerical experiments were run without any of the starting solutions. This consumed about 20 hours of CPU time for a 71 by 38 non-uniform grid; however, with a starter solution, based on a similar earlier case, the CPU times were reduced to half or even one third of the initial run time. The aforementioned efforts to cut computer time were carried out basically because of the extremely long

run times for the complicated case involving buoyancy effects.

The effects of the domain dependency were also investigated by increasing the lengths of the inlet and exit basal planes and also the y directional domain. The results once again confirmed that the existing domain was sufficient for the analysis.

5.7 Flow at the Trailing Edge of the Ice Sheet

The contours of the stream functions for the above cases show a tendency to drop rather steeply slightly before the end of the ice sheet. This numerical difficulty was improved by using a mirror image of the grid arrangement from the upstream half of the ice sheet at the downstream half . This ensured that the rather steep drop occured just at the edge of the ice sheet and also made the stream function smooth at the end of the ice sheet. The sheer drop of the stream function immediately after the ice sheet was in no way altered even after such a fine refinement of the trailing edge and this seemingly anomalous behaviour of the flow is explained as follows.

The v component of the velocity is zero at the inlet basal plane before the leading edge. The v velocity is a maximum at the leading edge and goes to a minimum down along the ice sheet towards its trailing edge. The v velocity is a function of the temperature gradient in the y direction at the ice sheet and since the water gets cooled by the ice sheet as it flows over it, the temperature gradient keeps dropping along the length of the ice sheet. The v velocity is set as zero at the exit basal plane . The u velocity is zero along the entire length of the ice sheet, to satisfy the no slip condition. The boundary condition for the stream function at the exit basal plane after the trailing edge took into account the

zero v velocity in that region. The stream function at the exit basal plane had been set as equal to that of the value at the trailing edge of the ice sheet , based on the stream function definition. This was the boundary condition that was fed into the Thomas algorithm as only either the value of a function or its gradient can be specified. More than one boundary condition for the stream function would overspecify the elliptic Poissonian equation relating the vorticity and the stream function(Roache). The stream function is allowed to have a gradient in the y direction, which means that a finite u velocity does exist at the node right after the trailing edge. This makes sense because the flow must have a u velocity after the trailing edge. This means that the flow is accelerating from a zero u velocity at the trailing edge to a finite velocity at the node that lies immediately after the trailing edge of the ice sheet. Continuity demands that this increased gradient of the u velocity be obtained at the expense of a decreased gradient of the v velocity. Also, the v velocity drops down to zero at the node next to the trailing edge from a finite velocity at the trailing edge and this results in the stream function near the trailing edge going through a steep drop . The above flow situation indicates clearly that the boundary conditions on the exit basal plane are critical in the modelling of the above flow situation.

5.8 Flow at Low Free Stream Velocities

The flow at lower free stream velocities resulted in the lowering of the inertial forces as compared to the buoyant forces and this resulted in the stream functions rising up to a larger extent on reaching the ice sheet as compared to those with higher free stream velocities.

5.9 CONCLUSIONS

The differentially heated square cavity, the laminar driven cavity and confirmation with Kuo's predictions have established the validity of the ADI scheme and the competence of the mathematical approach and the subsequent numerical modelling.

The main thrust of the investigation , namely to find out whether the recirculation cells exist for the case of flow of cold water over an horizontal ice sheet and to investigate the oscillatory nature of the solution at low free stream velocities has led to the following conclusions.

(1) No recirculations were observed at the free stream velocities of 0.02 m/s , which was the velocity for recirculation cells to have appeared in Wilson and Sarma's [18] investigation.

(2) The oscillations of the residuals were due to the extrapolative boundary conditions at the edge of the ice sheet where an infinitely long sheet was modelled.

(3) Furthermore , the wavy patterns in the stream functions do bear the telltale signs of the spatial oscillations in the flow situation, manifesting themselves as wiggles , which was corrected by upwinding difference scheme. Roache [12] has shown that the upwinding difference scheme used by Gosman et al.,[7] is infact an arithmetic equivalent of an upwind scheme that has got a superior truncation error adjustment as compared to the technique used in this scheme . This may have led to the formation of recirculation cells and wiggles in Wilson and Sarma's [18] investigation.

5.10 Recommendations

An experimental investigation for the above problem is suggested with the stream function contours being ascertained by non invasive techniques such as using Thymol blue and other Ph indicators and using timed sequential photographs as carried out by Wilson and Vyas [19]. This data could be compared with the numerical investigations based on the upwinding scheme used in this work, the second upwinding method or the donor cell method as explained by Roache [12] and comparing the accuracy of the computations.

The central differencing scheme could also be combined with one or both of the above mentioned upwinding schemes so that the accuracy is improved. This could be carried out by using a weighted average of these schemes after going through one iterative step in every scheme to determine the optimal combination that would damp out the oscillations produced due to central differencing scheme and also give a result that is more accurate.

REFERENCES

1. Carey, V. P., and Gebhart B., Visualisation of the flow adjacent to a vertical ice surface melting in cold pure water, J. Fluid Mech., 107, 37-55 (1981) .

2. Carl E. Pearson, A computational method for viscous flow problems, J. Fluid Mech., 21, 611-622 (1965).

3. Carrier G.F., and Lin C.N., On the nature of the boundary layer near the leading edge of a flat plate, Quart. Appl. Math., 6,63-68, (1948).

4. Creare Inc., Fluent Manual , Version 2.9 update, Hanover, New Hampshire, 1987 .

5. Fanning A.E., and Mueller T.J., Numerical and experimental investigation of the oscillating wake of a blunt based body, AIAA Journal 11, 1486-1491 .-1491 .

6. Gebhart B., Jaluria Y., Mahajan R.L., and Sammakia B., Buoyancy Induced flow and transport, Hemisphere publishing corporation, 1988 .

7. Gosman A. D., Pun W.M., Runchal A.K., Spalding D.B., Wolfshtein M., Heat and mass transfer in recirculating flows , Academic Press, London and New York (1969).

8. Kuo Y.H., On the flow of incompressible viscous fluid past a flat plate at moderate Reynolds numbers, J. Math. Phys. 32, 83-101 (1954) .

9. Lee, J.J., Melting of a vertical ice wall by natural convection into pure or saline water, M.A.Sc Thesis, Memorial university of Newfoundland, Newfoundland (1979) .

10. Lin,D.S., Natural Convection Heat Transfer in vertical annular and rectangular enclosures containing water near its density maximum, Ph.D. Thesis, University of Pennsylvania, Pennsylvania (1986).

11. Lin, D.S., and Nansteel M.W., Natural Convection Heat Transfer in a square enclosure containing water near its density maximum, Int. J. Heat Mass Transfer 36, 2319-2329 (1987) .

12. Roache P.J., Computational Fluid Dynamics, Hermosa publishers (1985) .

13. Roache P.J., and Mueller T.J., Numerical solutions of laminar seperated flows, AIAA Journal 8, 530-538 (1970).

14. Samuels M.R., and Churchill S.W., Stability of a fluid rectangular region heated from below , A.I.CH.E. Journal 13, 77-85 .

15. Schlichting H., Boundary layer theory , McGraw Hill book company, New York, 1968 .

16. Srivatsava P.K., Buoyancy effects on heat , mass and momentum transfer during the melting of a horizontal ice sheet above fresh or saline water flowing laminar Reynolds numbers, M.A.Sc. Thesis, Memorial university of Newfoundland, Newfoundland (1979).

17. Wilson N.W. and Lee J.J., Melting of a vertical ice wall by free convection into fresh water, J. Heat Transfer, 103, 13-17 (1981) .

18. Wilson N.W. and Sarma T.S., Buoyancy effects on heat, mass and momentum transfer during the melting of a horizontal ice surface below saline water flowing at laminar Reynolds numbers, private communication

19. Wilson N.W. and Vyas B.D., Velocity profiles near a vertical ice surface melting into fresh water J. Heat Transfer, 101, 313-317 (1979).

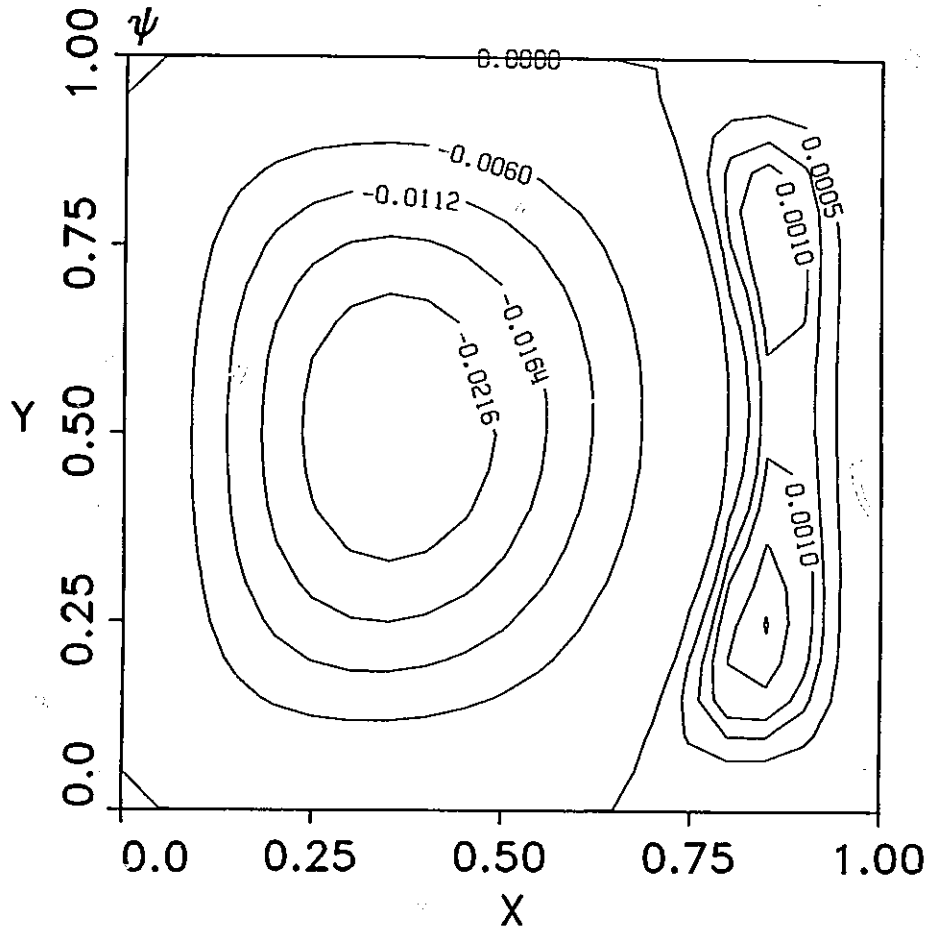


Fig 1 : Contours of dimensionless Stream Function for Ra = 1000, R = 0.4 and a 21 by 21 uniform grid

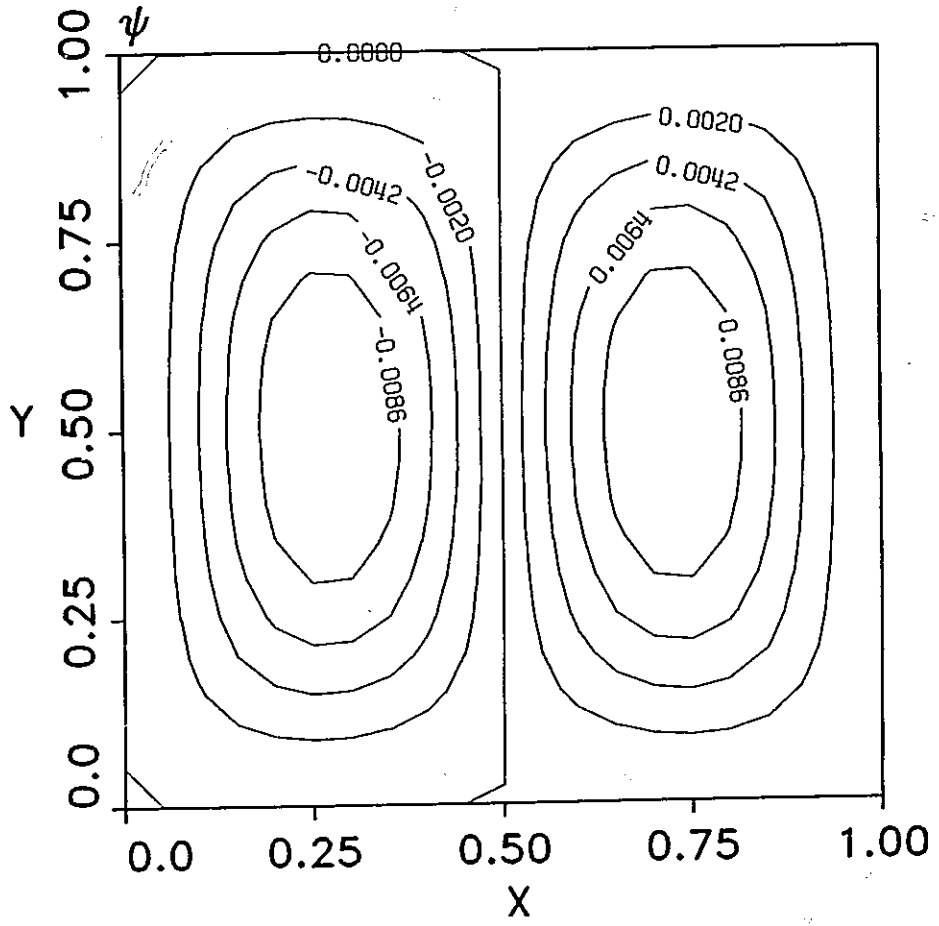


Fig 2 : Contours of dimensionless Stream Function for Ra = 1000, R = 0.5 and a 21 by 21 uniform grid

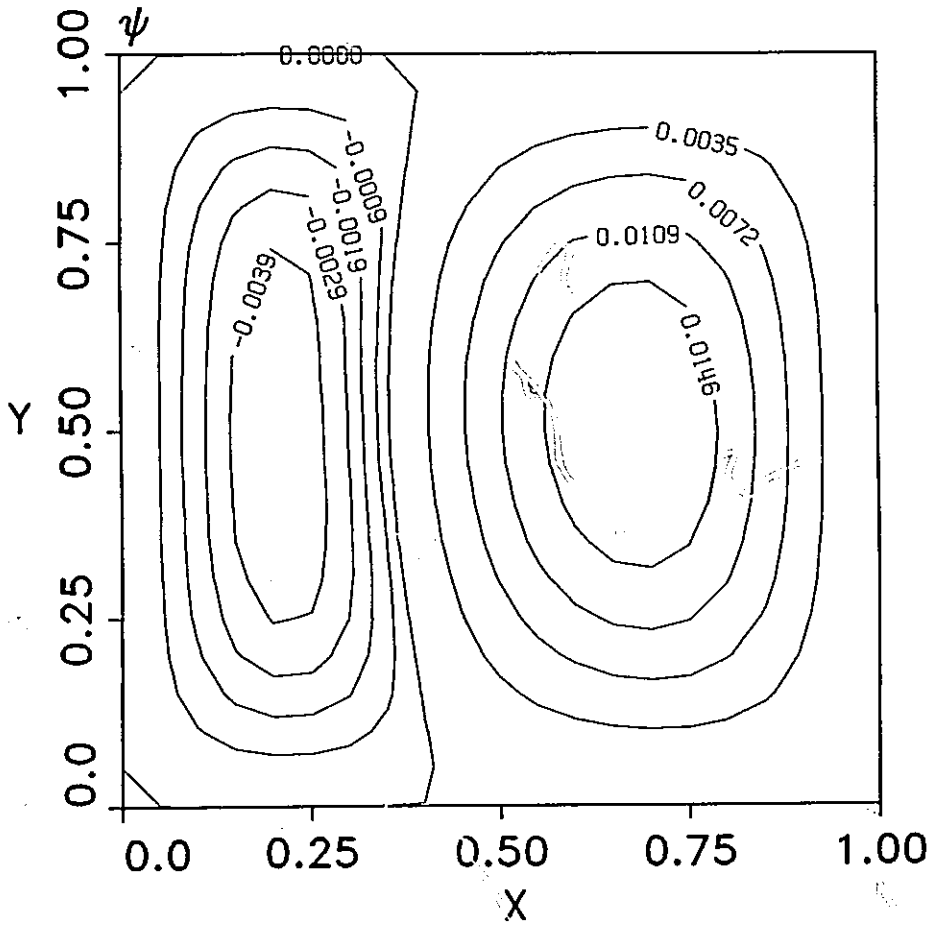


Fig 3 : Contours of dimensionless Stream Function for Ra = 1000, R = 0.55 and a 21 by 21 uniform grid

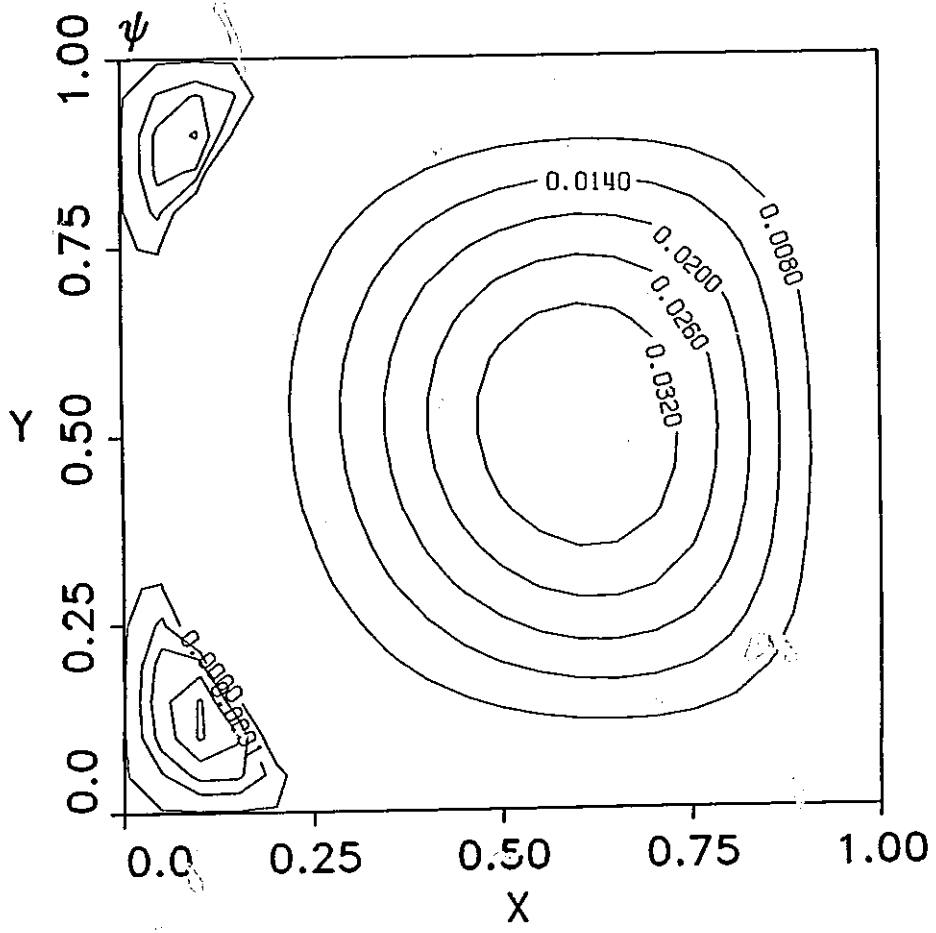


Fig 4 : Contours of dimensionless Stream Function for Ra = 1000, R = 0.67 and a 21 by 21 uniform grid

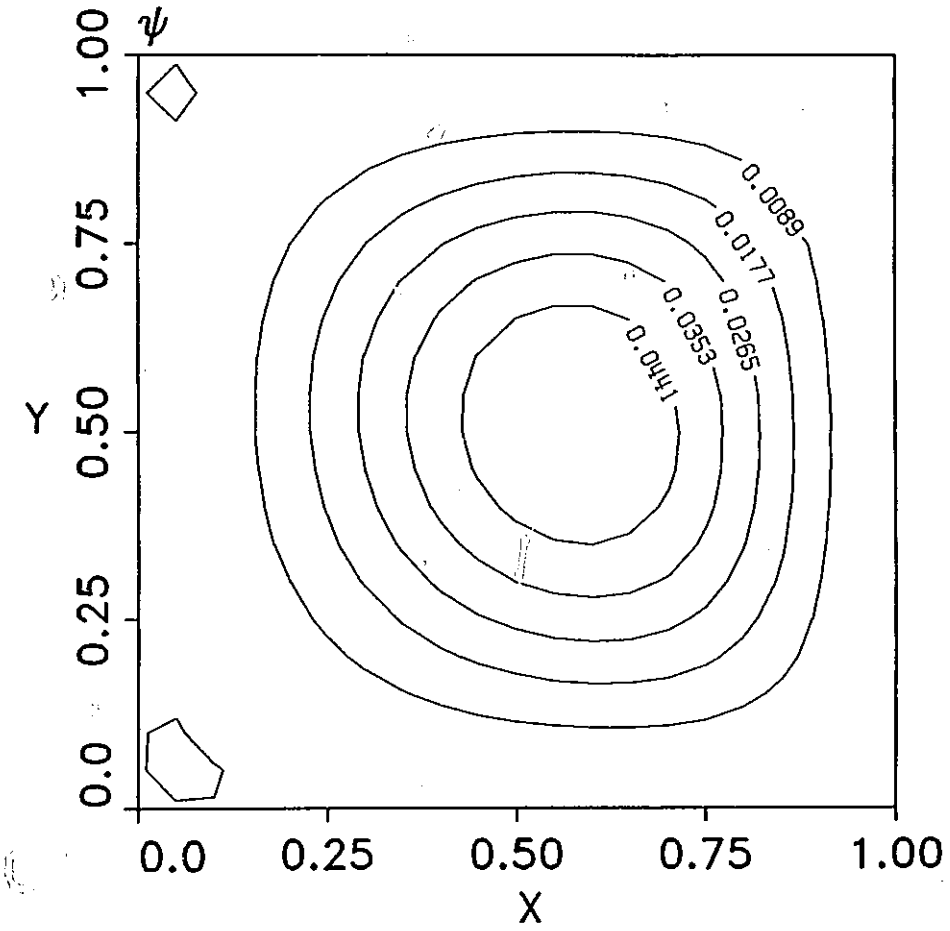


Fig 5 : Contours of dimensionless Stream Function for Ra = 1000, R = 0.75 and a 21 by 21 uniform grid

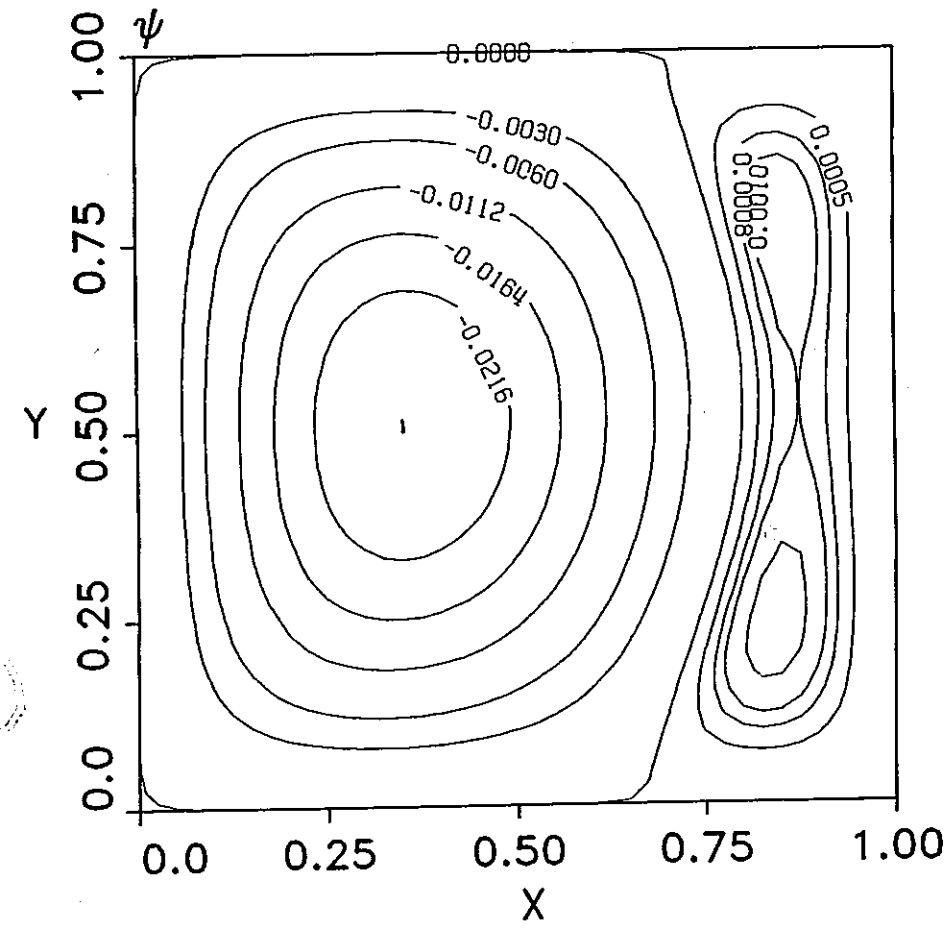


Fig 6 : Contours of dimensionless Stream Function for Ra = 1000, R = 0.4 and a 41 by 41 uniform grid

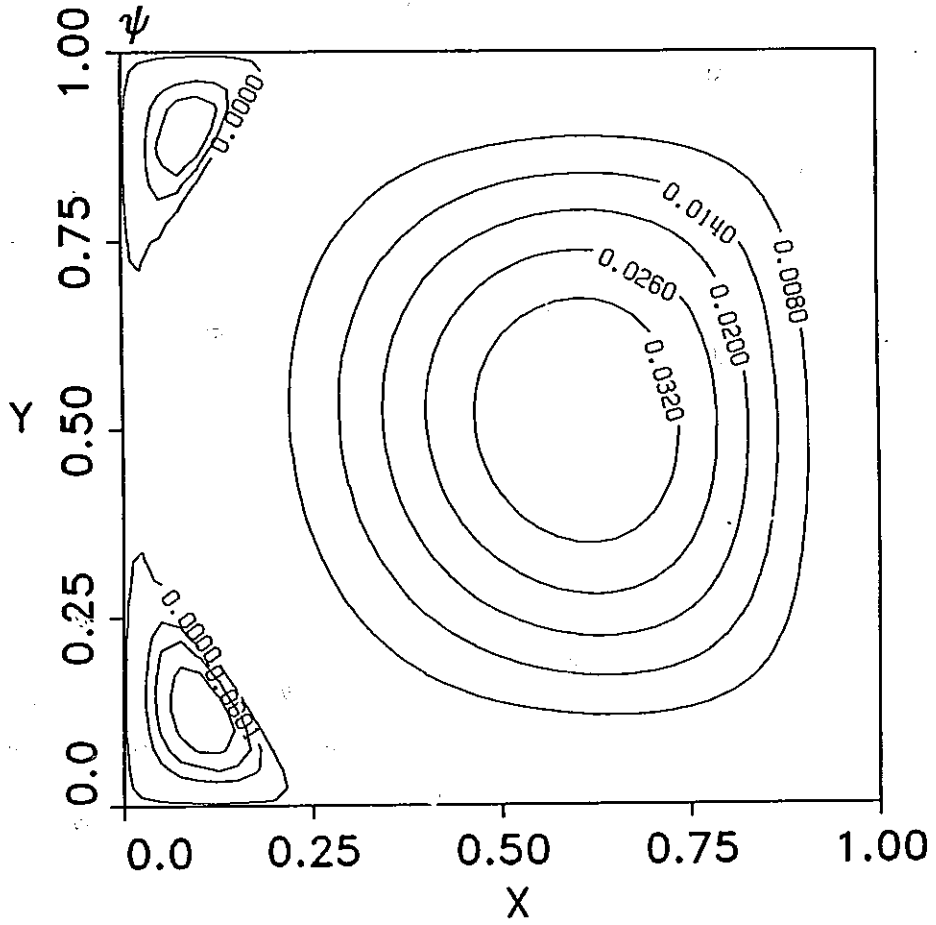


Fig 7 : Contours of dimensionless Stream Function for Ra = 1000, R = 0.67 and a 41 by 41 uniform grid

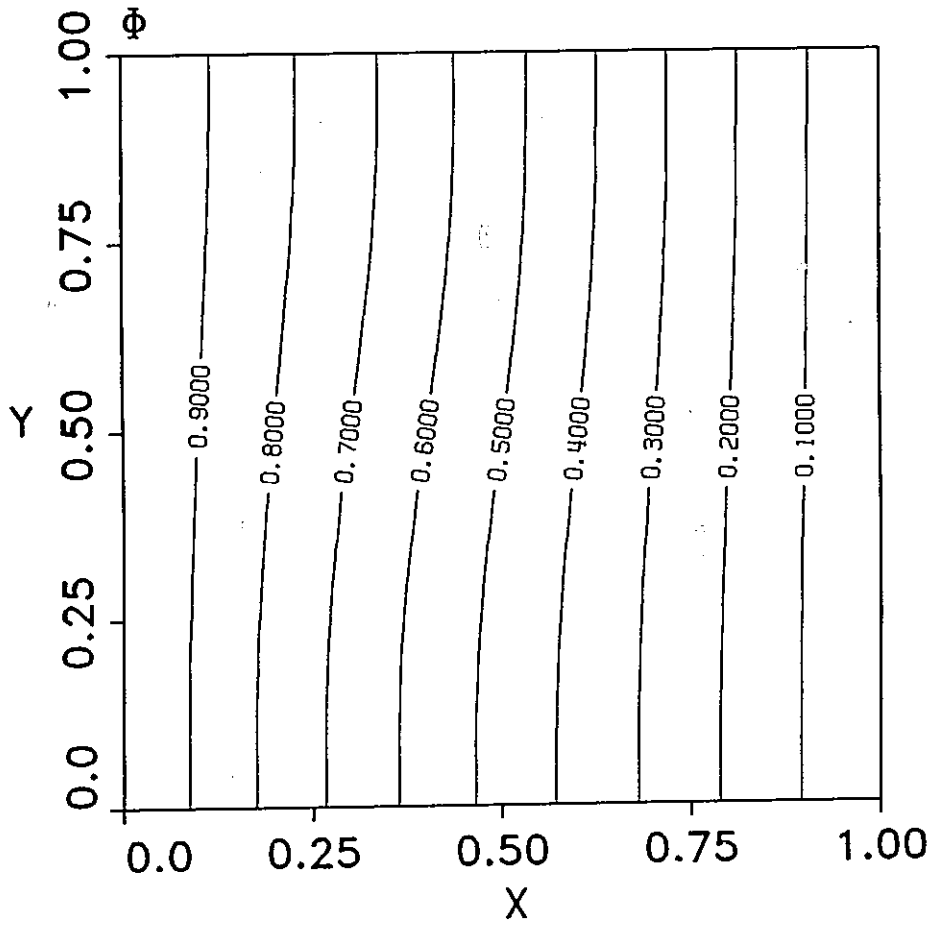


Fig 8 : Contours of dimensionless Temperature for Ra = 1000, R = 0.4 and a 21 by 21 uniform grid

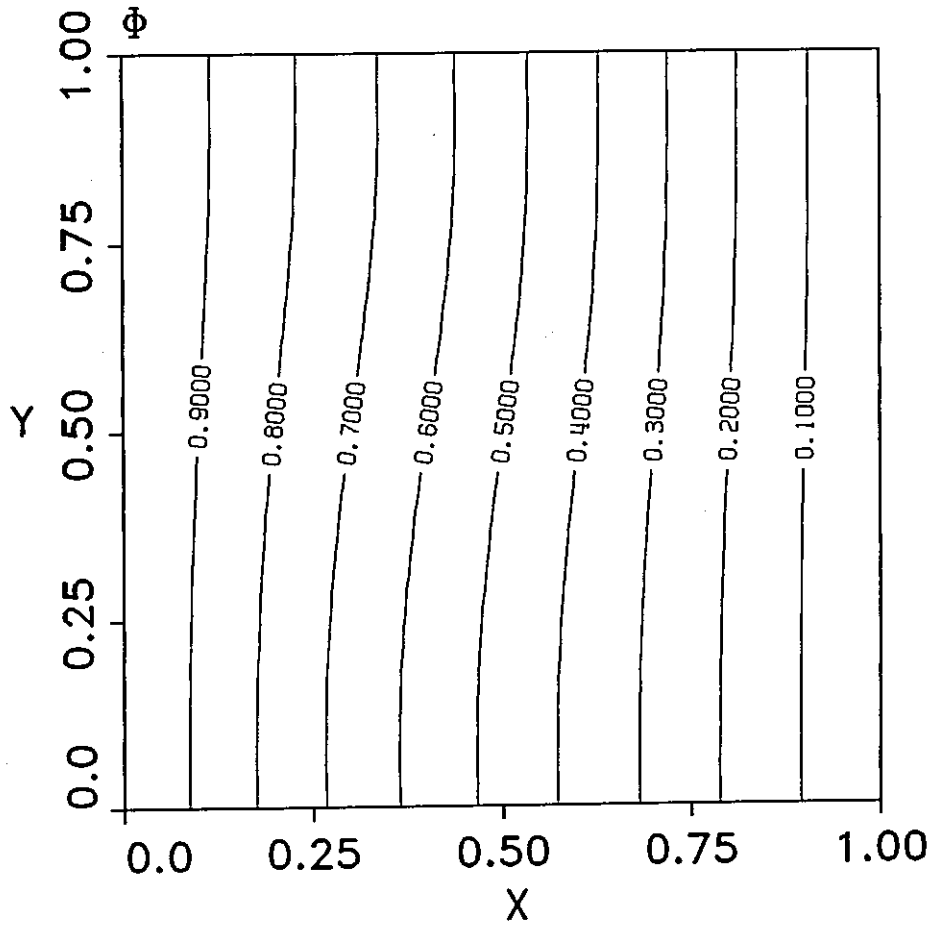


Fig 9 : Contours of dimensionless Temperature for Ra = 1000, R = 0.4 and a 41 by 41 uniform grid

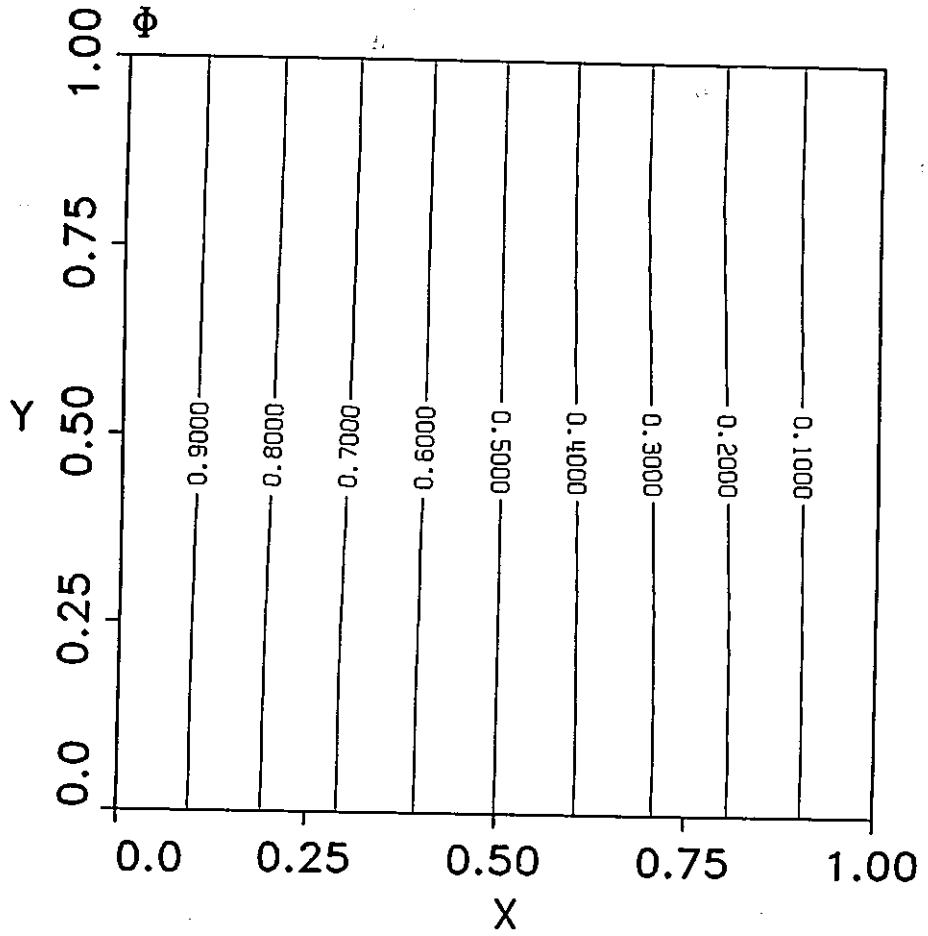


Fig 10 : Contours of dimensionless Temperature for Ra = 1000, R = 0.5 and a 21 by 21 uniform grid

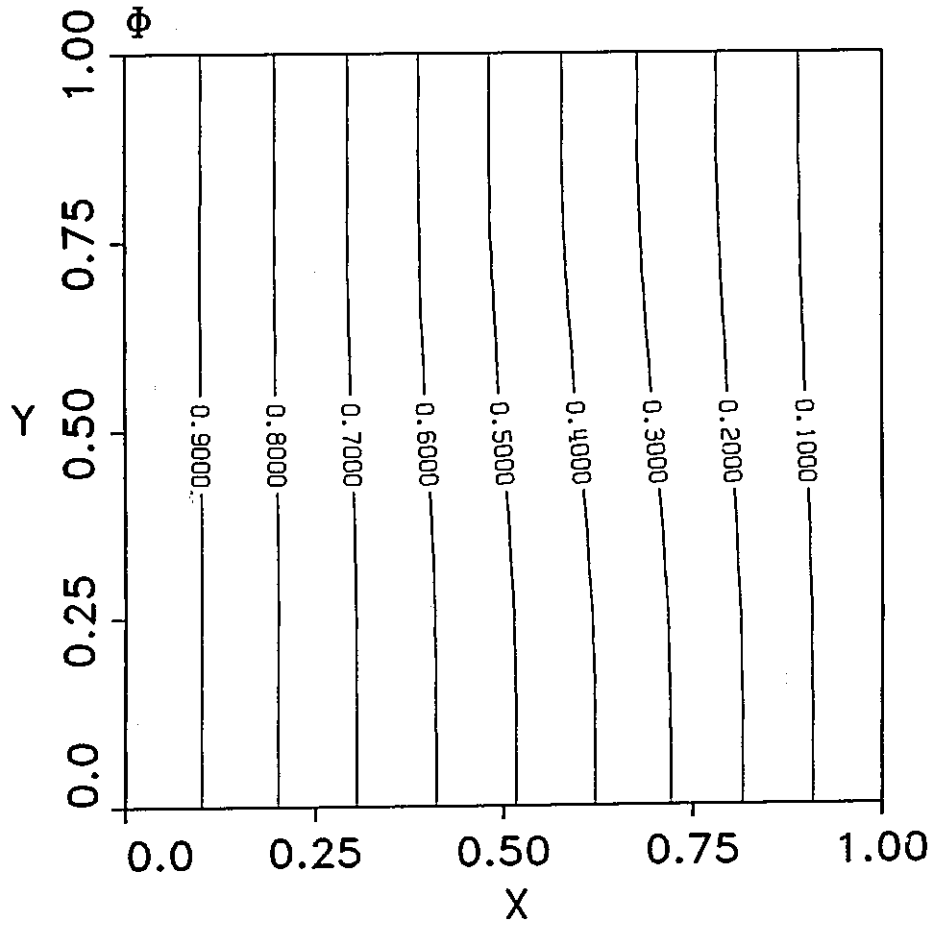


Fig 11 : Contours of dimensionless Temperature for Ra = 1000, R = 0.55 and a 21 by 21 uniform grid

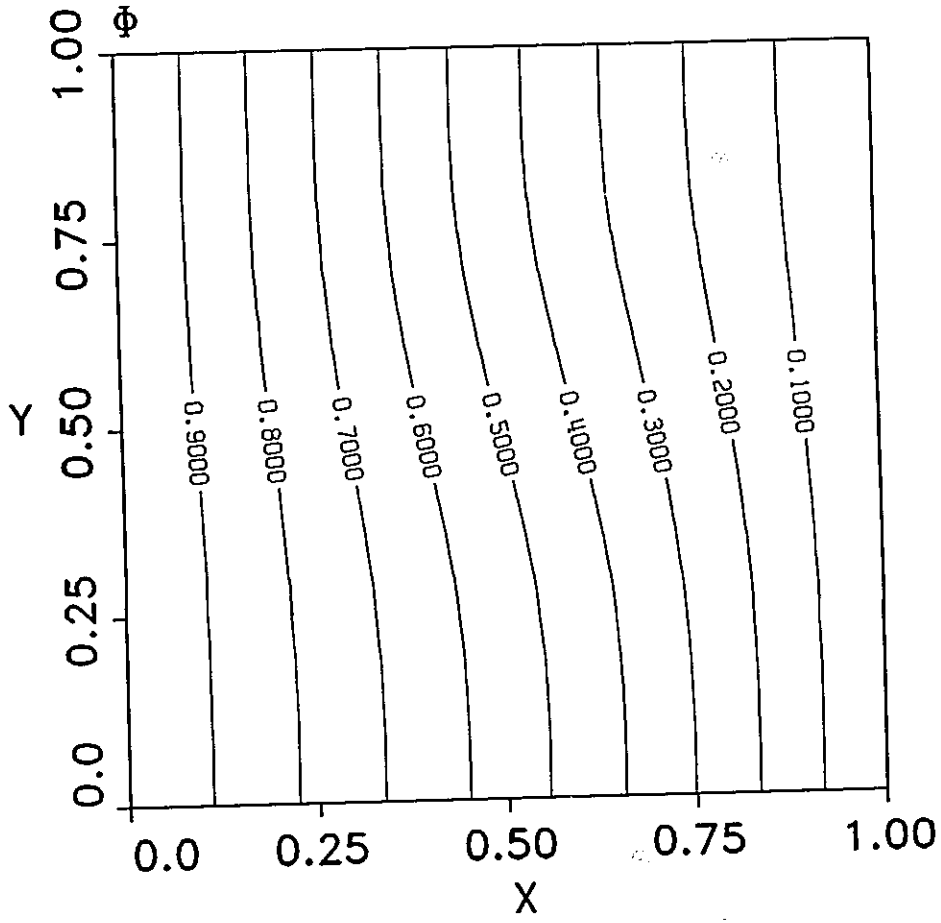


Fig 12 : Contours of dimensionless Temperature for Ra = 1000, R = 0.67 and a 21 by 21 uniform grid

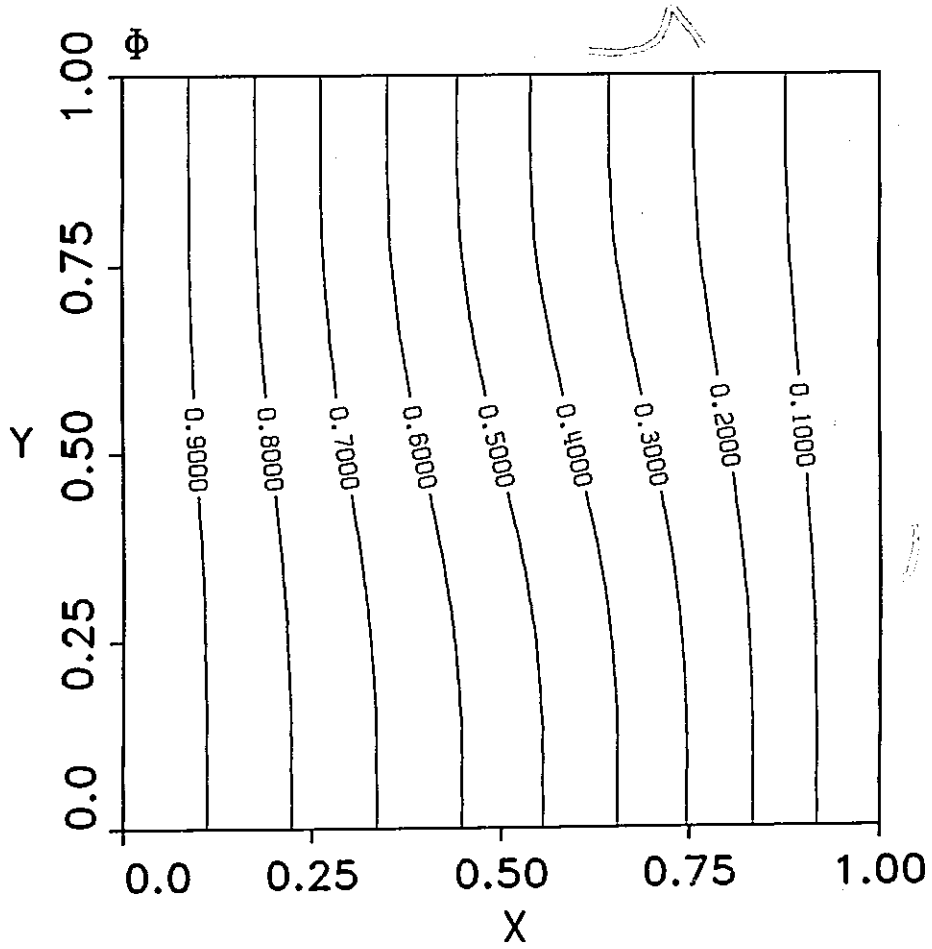


Fig 13 : Contours of dimensionless Temperature for Ra = 1000, R = 0.67 and a 41 by 41 uniform grid

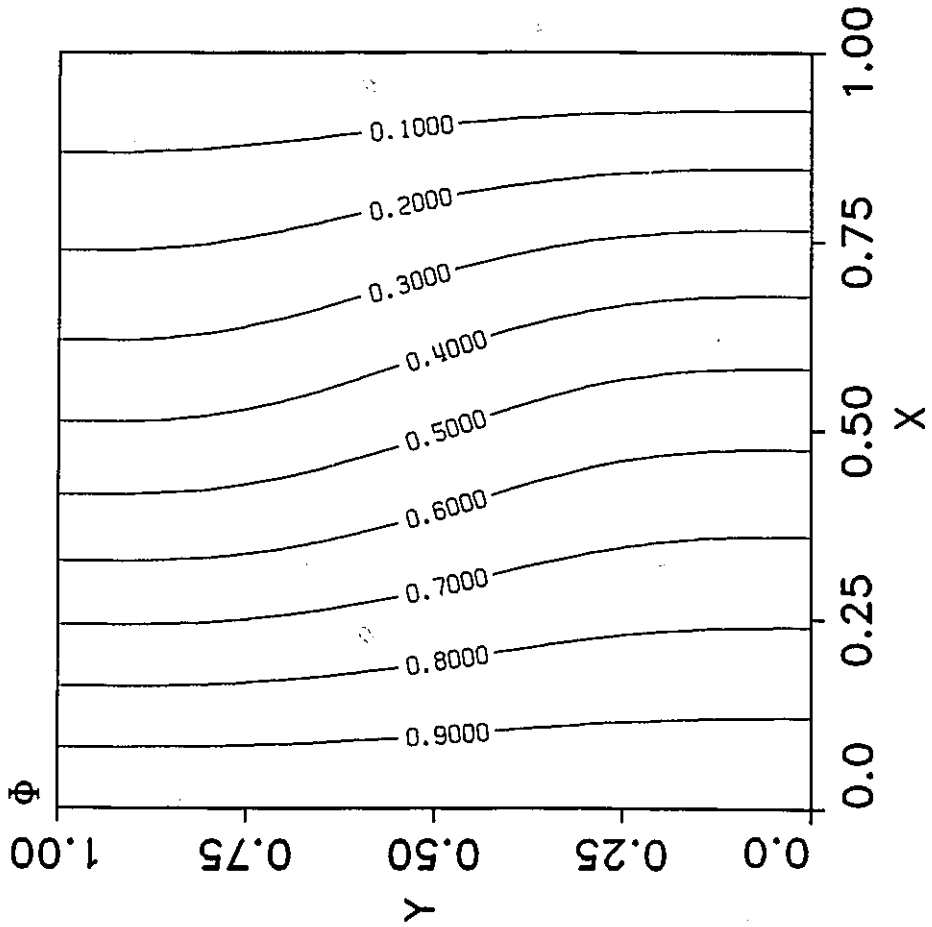


Fig 14 : Contours of dimensionless Temperature for Ra = 1000, R = 0.75 and a 21 by 21 uniform grid

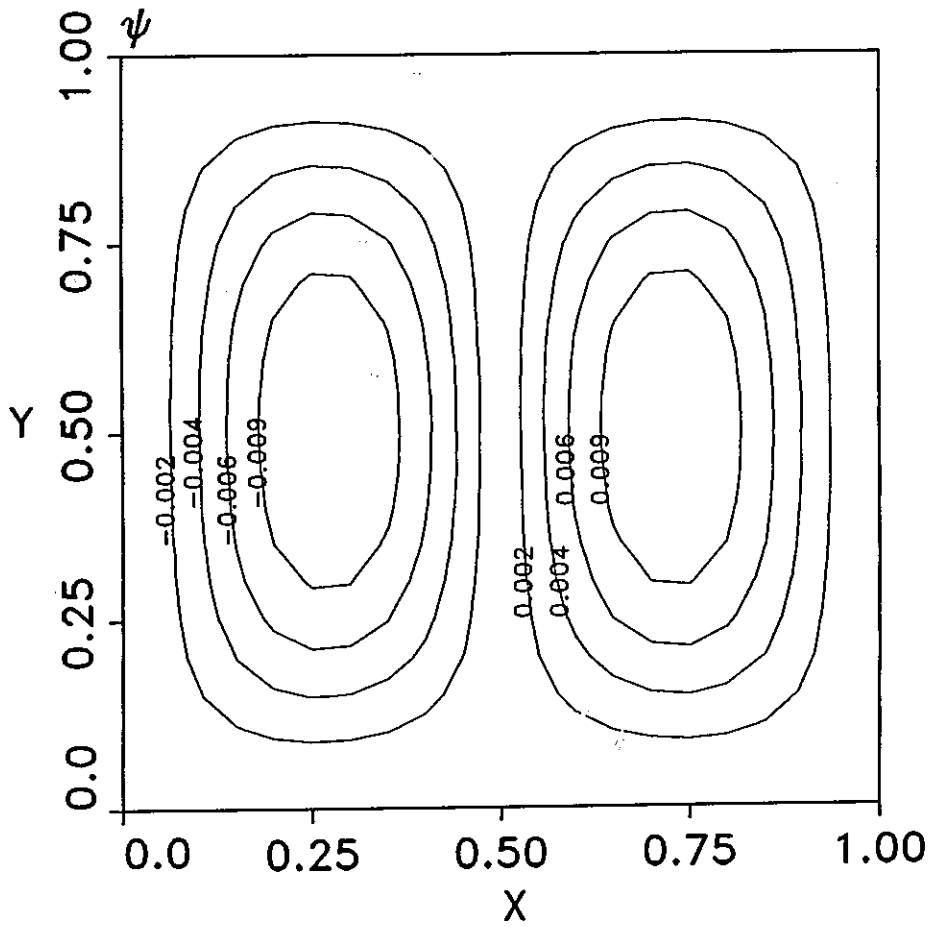


Fig 15 : Contours of dimensionless Stream Function for Ra = 1000, R = 0.5 and a 25 by 25 non-uniform grid

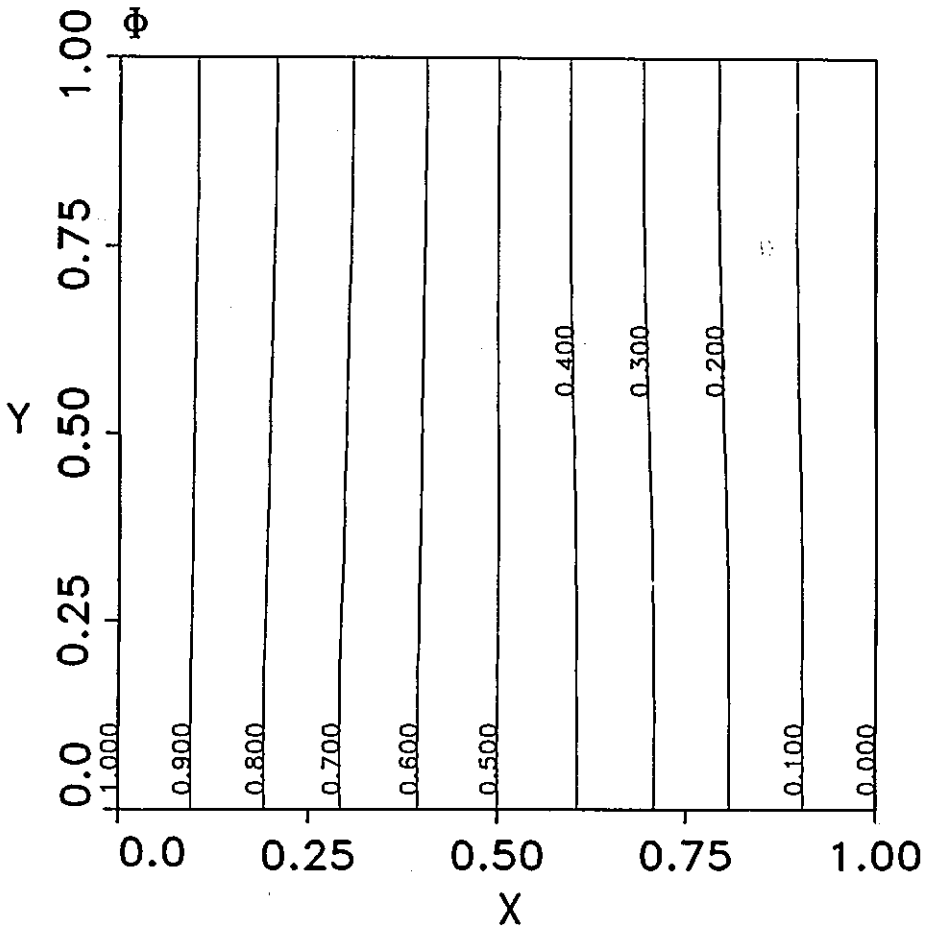


Fig 16 : Contours of dimensionless Temperature for Ra = 1000, R = 0.5 and a 25 by 25 non-uniform grid

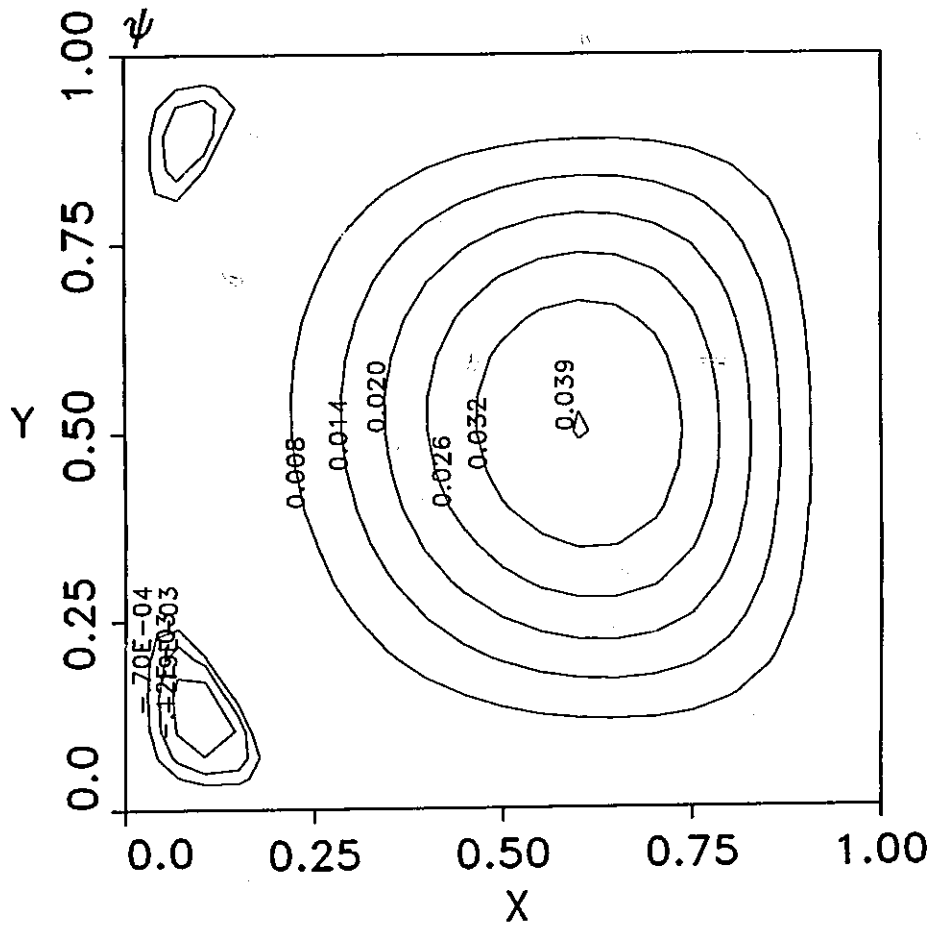


Fig 17 : Contours of dimensionless Stream Function for Ra = 1000, R = 0.67 and a 25 by 25 non-uniform grid

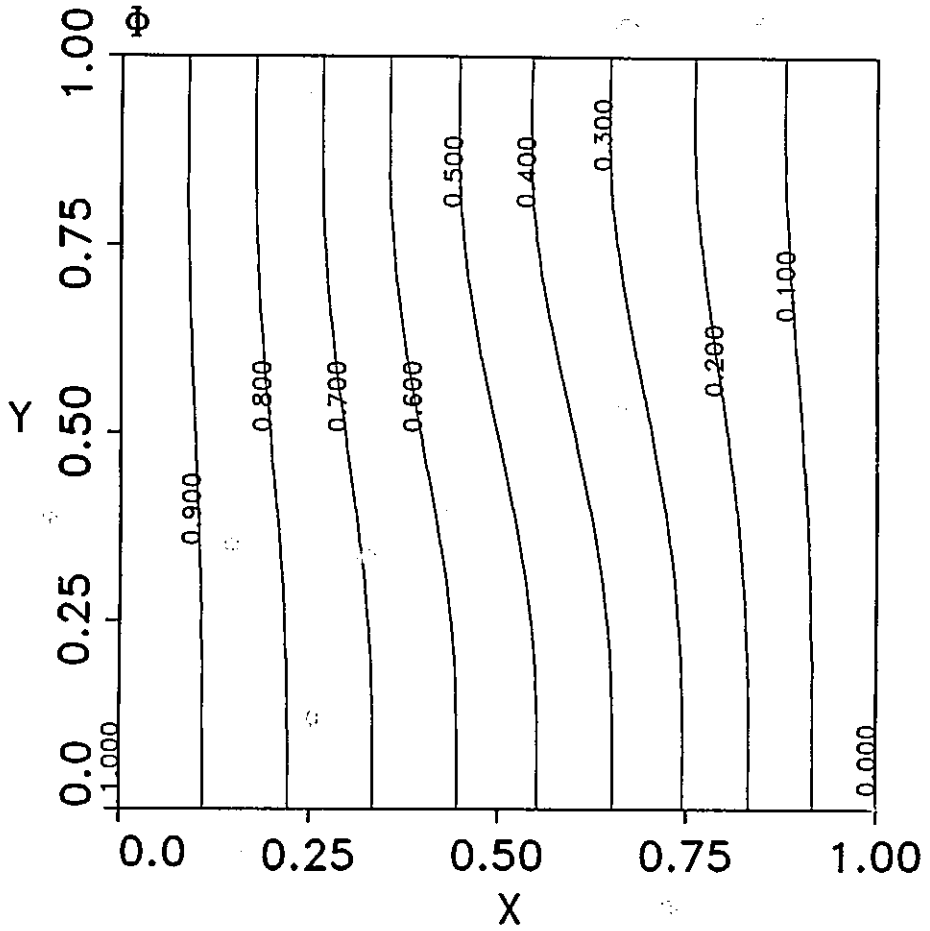


Fig 18 : Contours of dimensionless Temperature for Ra = 1000, R = 0.67 and a 25 by 25 non-uniform grid

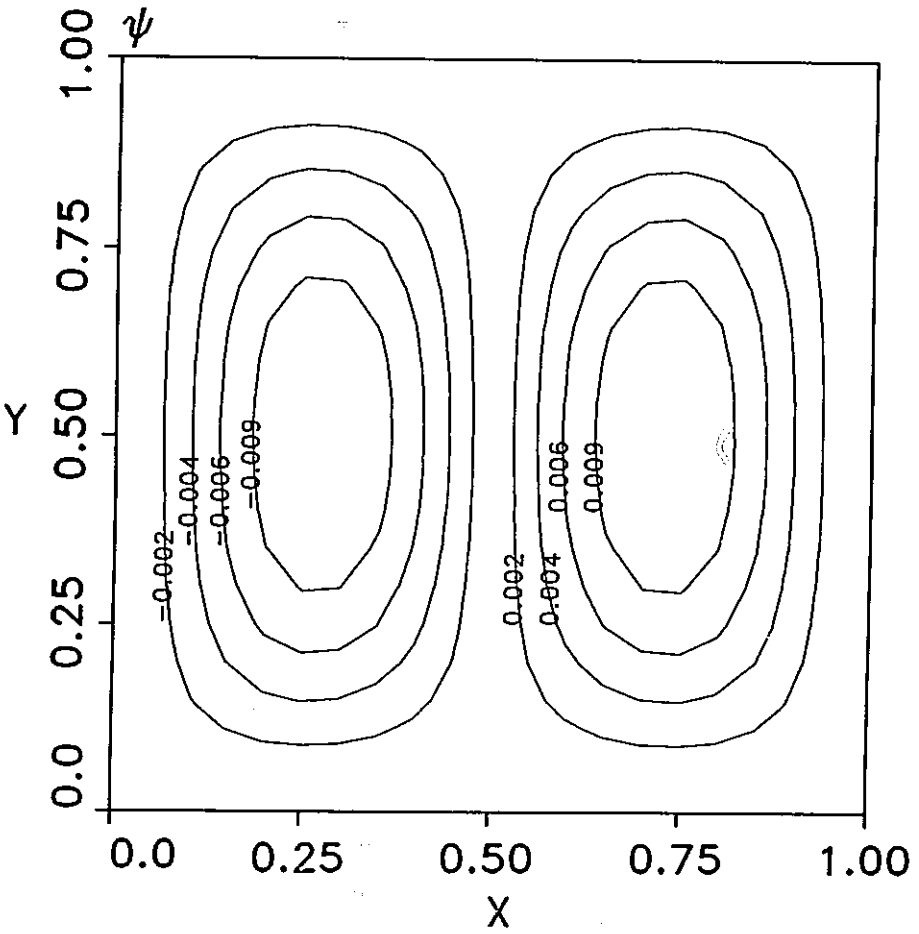


Fig 19 : Contours of dimensionless Stream Function for Ra = 1000, R = 0.5 and a 29 by 29 non-uniform grid

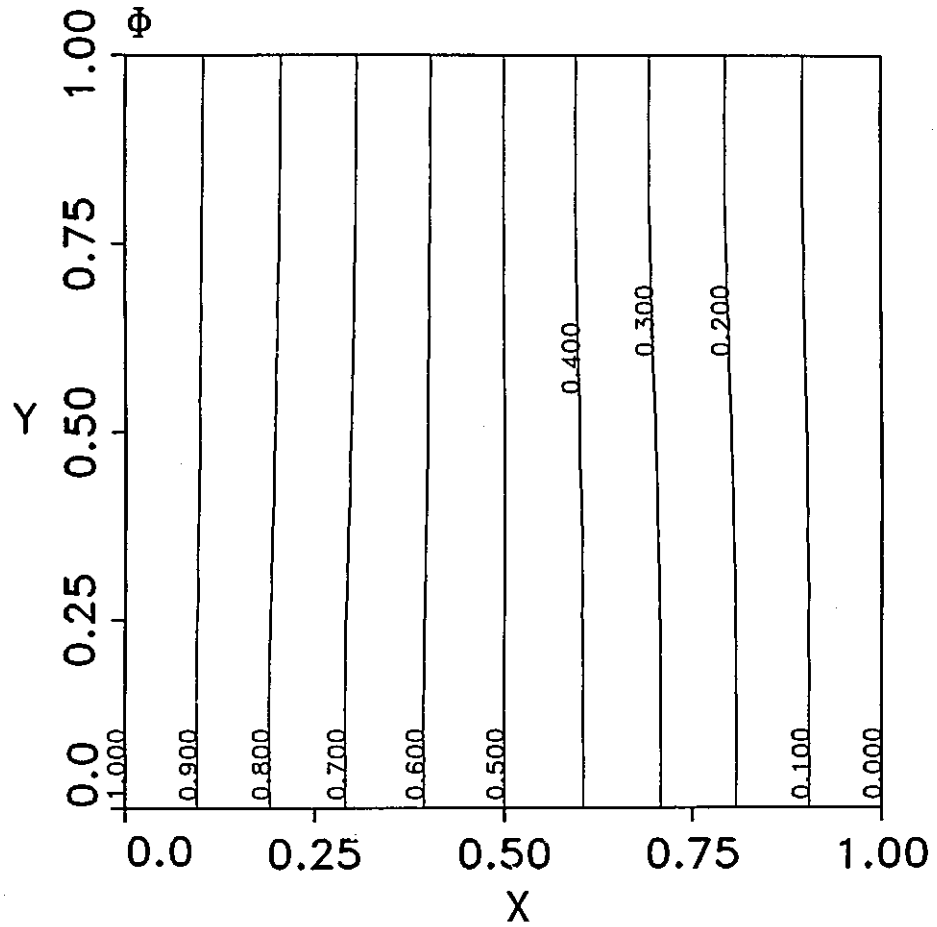


Fig 20 : Contours of dimensionless Temperature for Ra = 1000, R = 0.5 and a 29 by 29 non-uniform grid

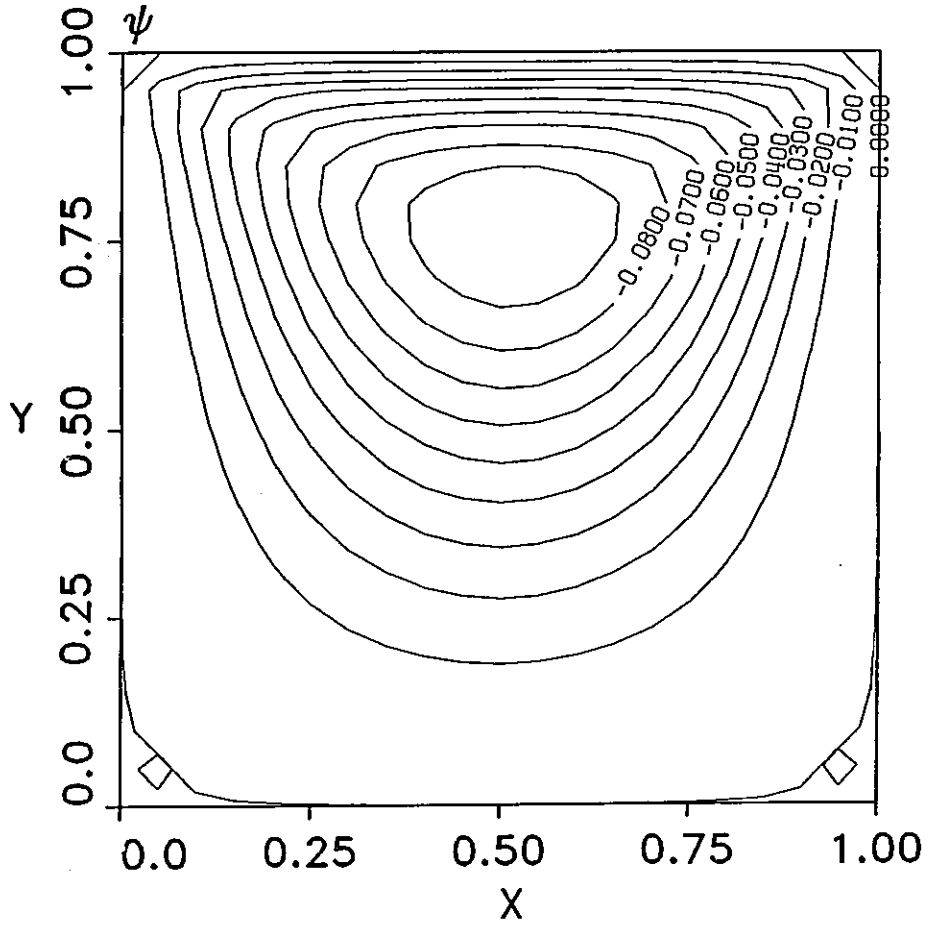


Fig 21 : Contours of dimensionless Stream Function for Re = 10 and a 21 by 21 uniform grid

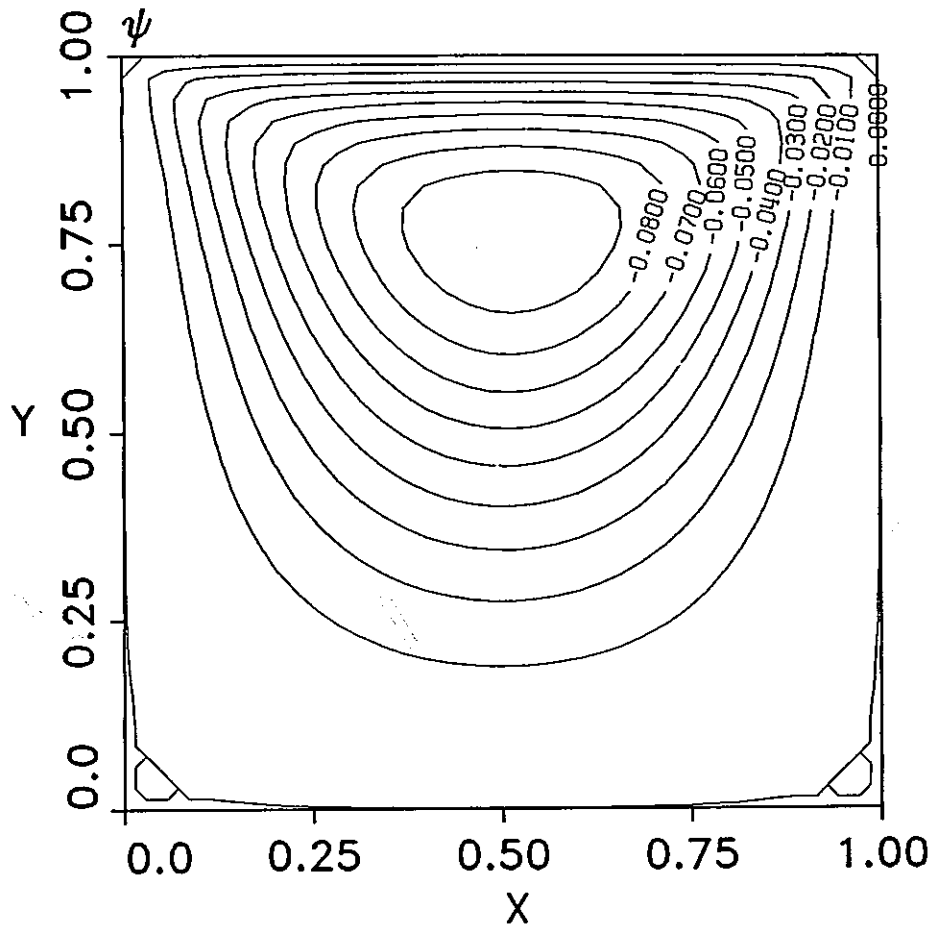


Fig 22 : Contours of dimensionless Stream Function for Re = 10 and a 36 by 36 uniform grid

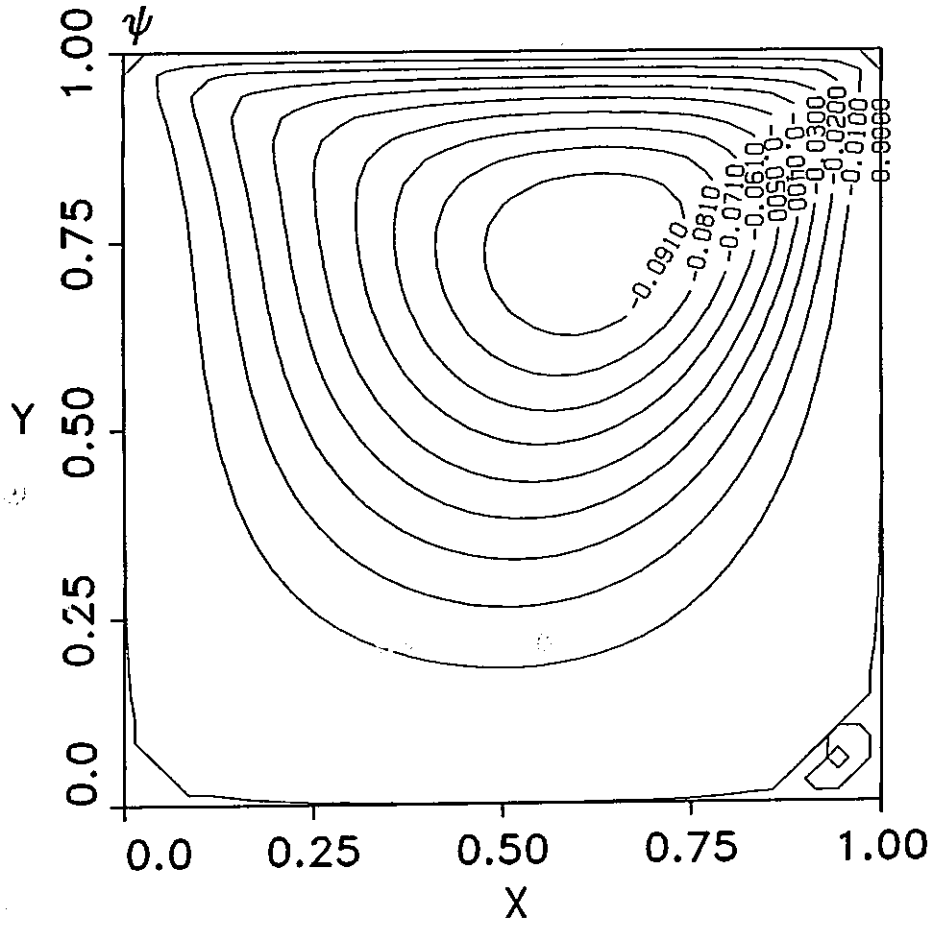


Fig 23 : Contours of dimensionless Stream Function for Re = 100 and a 36 by 36 uniform grid

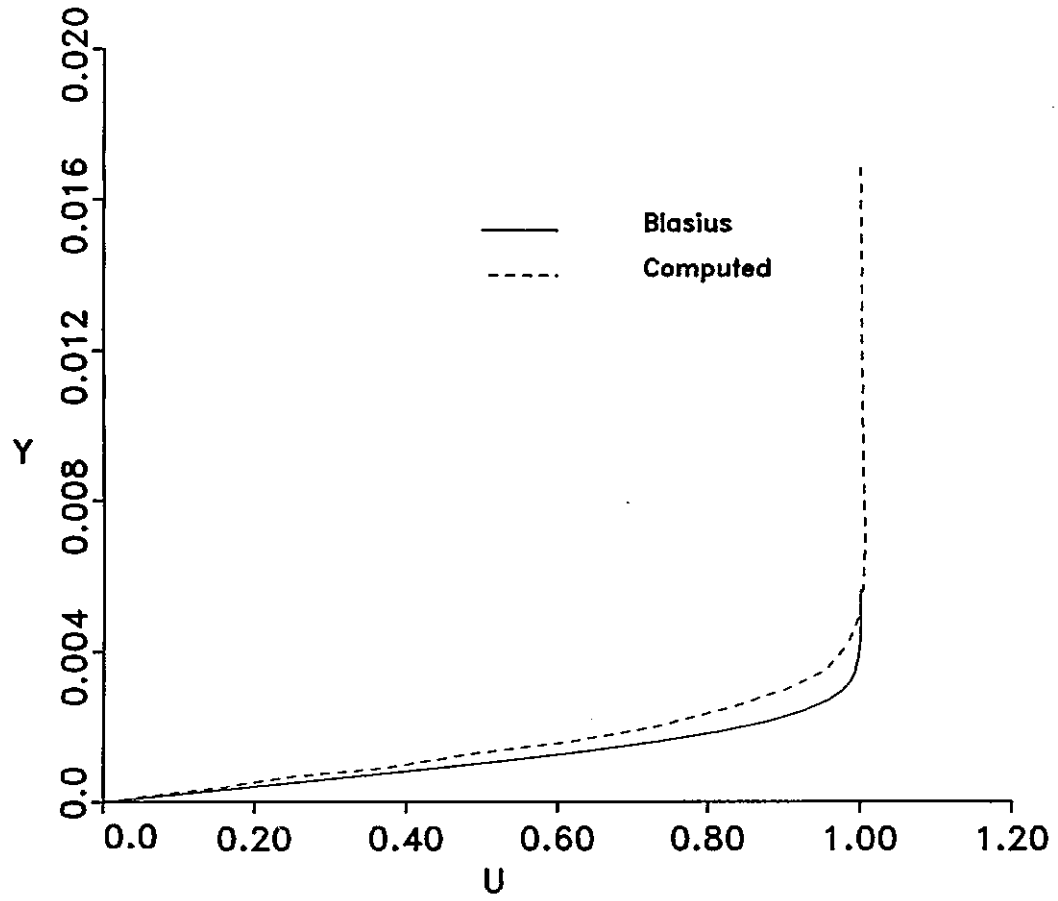


Fig 24 : Comparison of the normalised computed u velocity profile with the Blasius profile at x = 0.0258

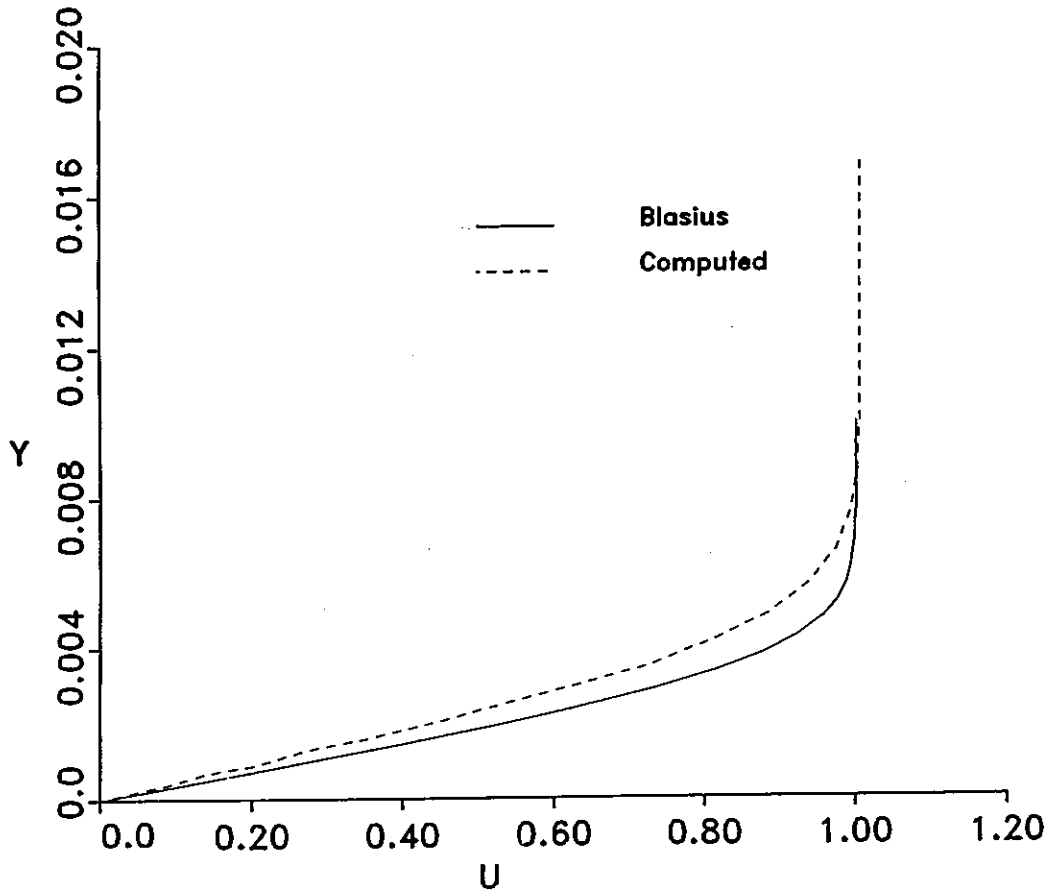


Fig 25 : Comparison of the normalised computed u velocity profile with the Blasius profile at x = 0.0814

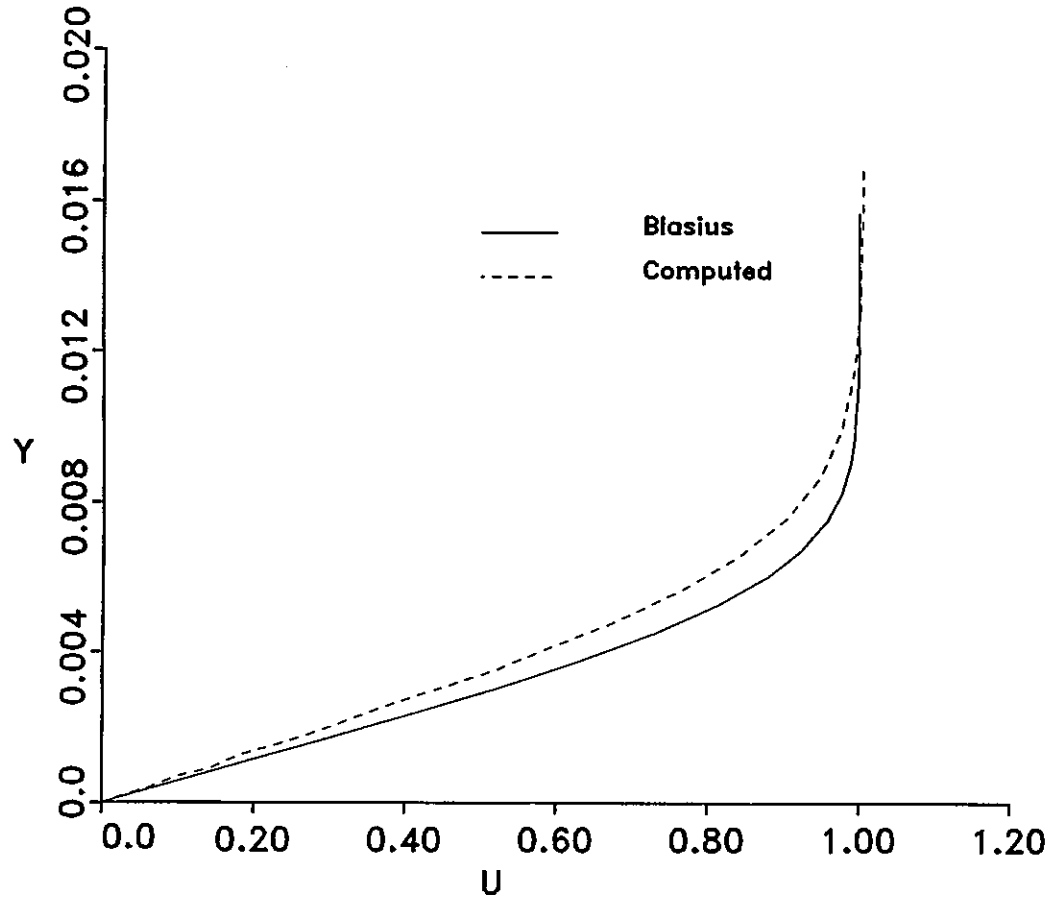


Fig 26 : Comparison of the normalised computed u velocity profile with the Blasius profile at x = 0.2014

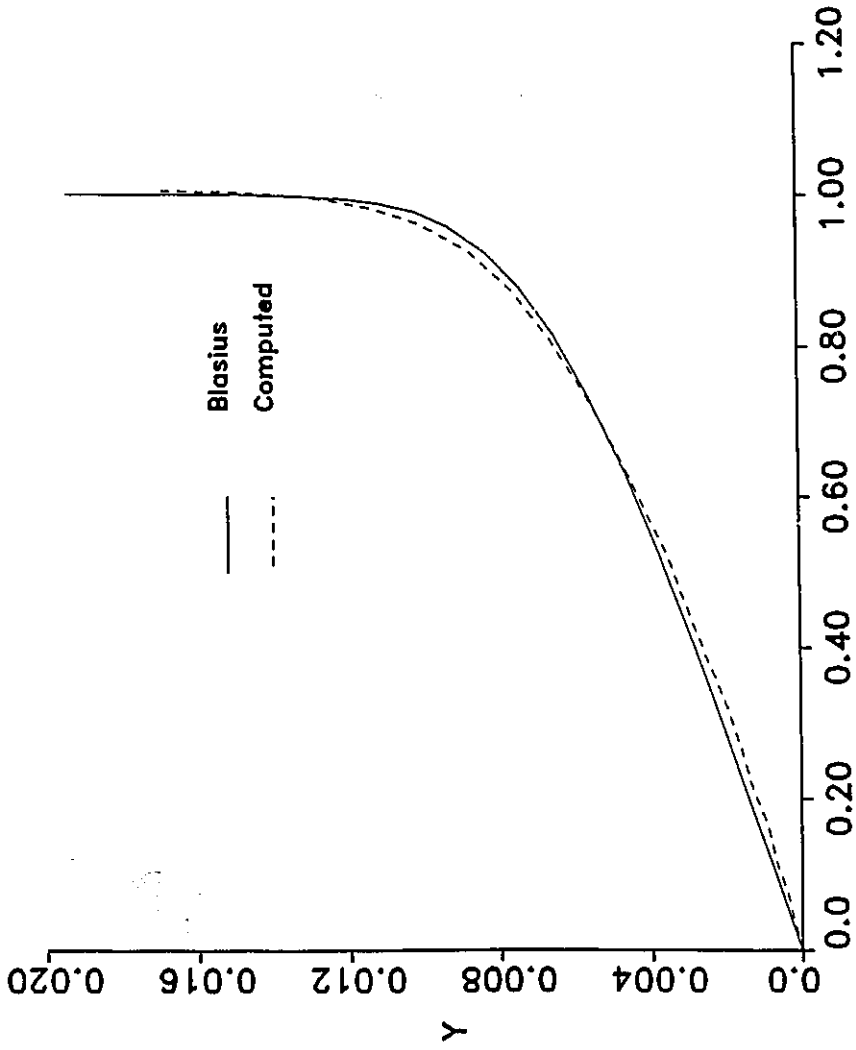


Fig 27 : Comparison of the normalised computed u velocity profile with the Blasius profile at x = 0.3063

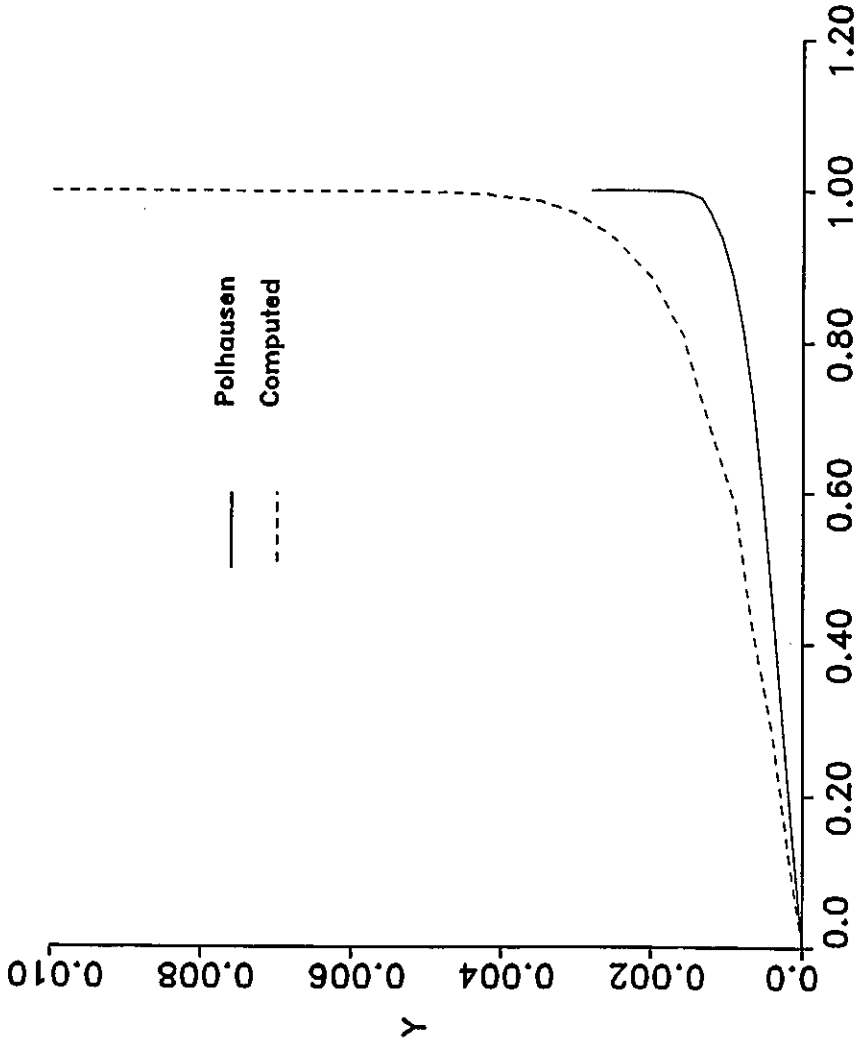


Fig 28 : Comparison of the dimensionless temperature profile with the Polhausen profile at x = 0.0258

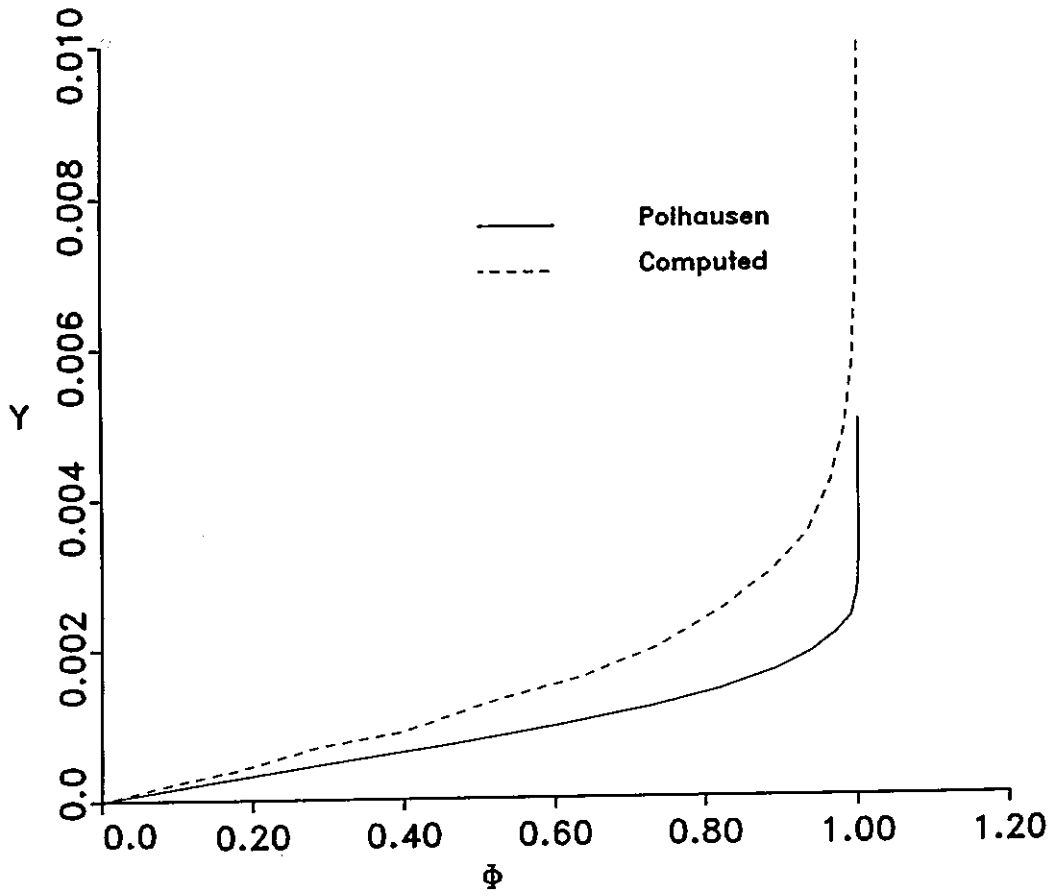


Fig 29 : Comparison of the dimensionless temperature profile with the Polhausen profile at x = 0.0814

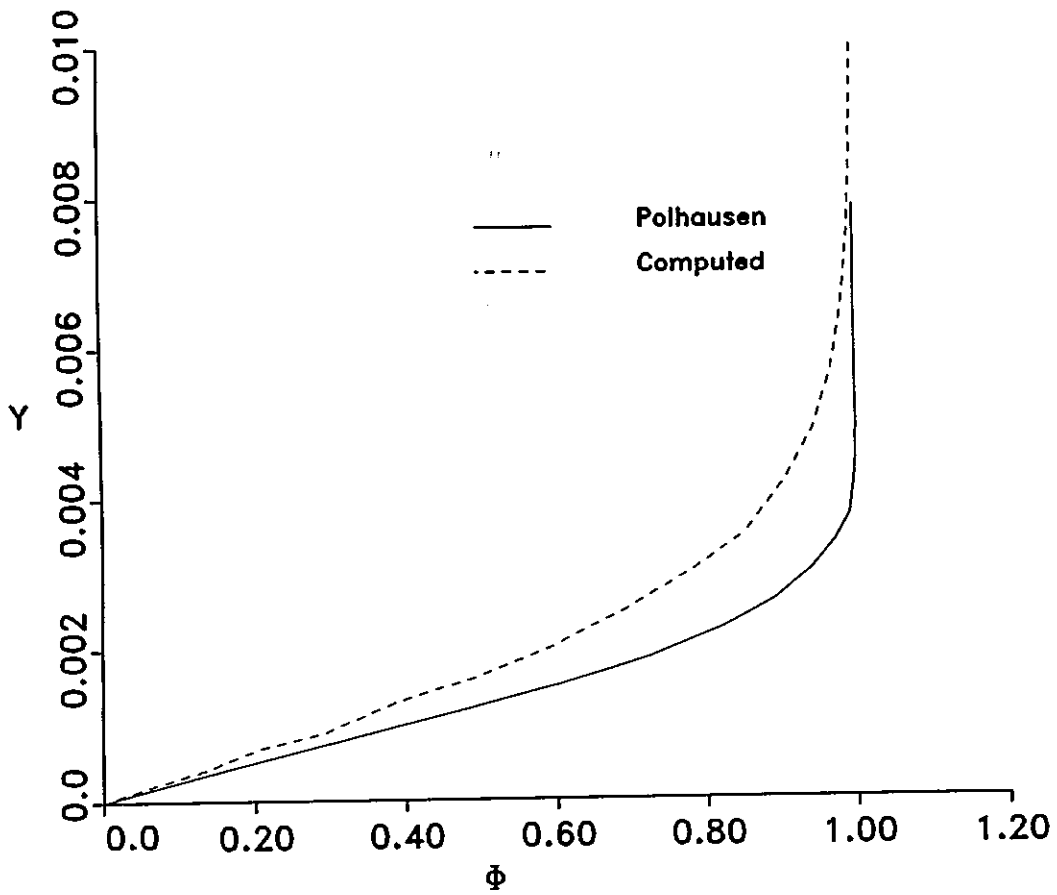


Fig 30 : Comparison of the dimensionless temperature profile with the Polhausen profile at x = 0.2014

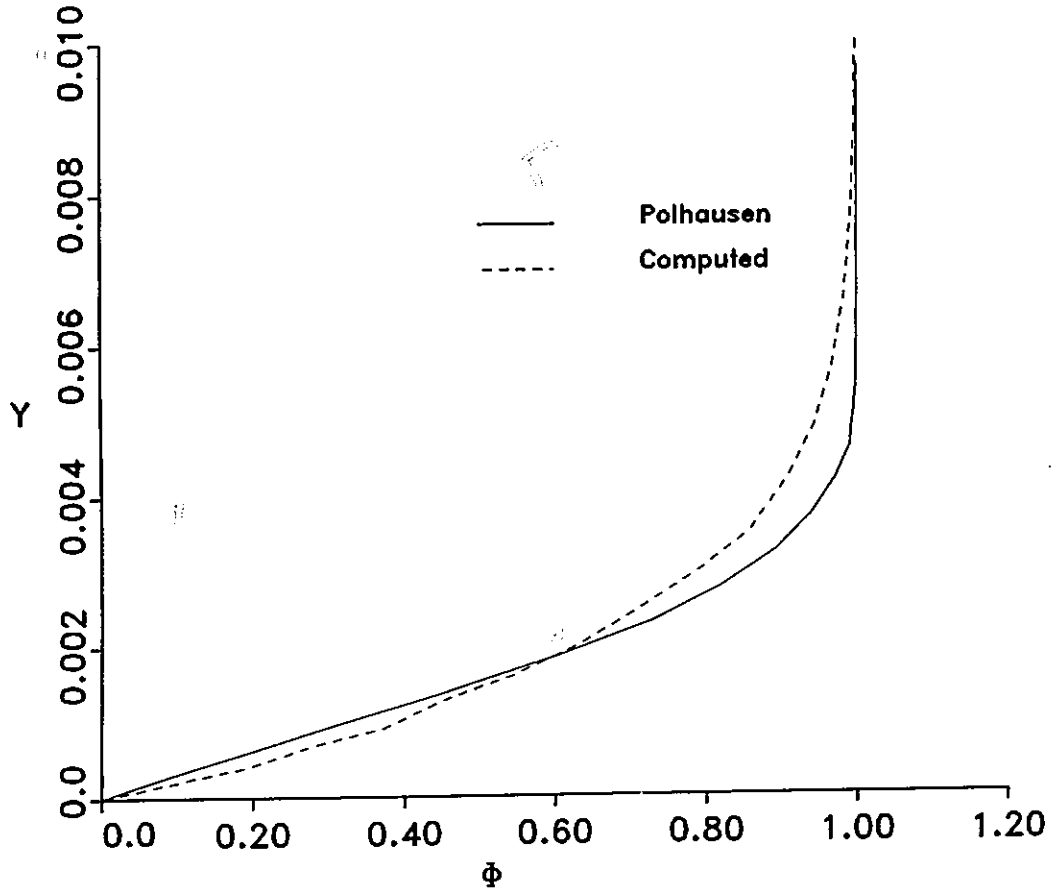


Fig 31 : Comparison of the dimensionless temperature profile with the Polhausen profile at x = 0.3063

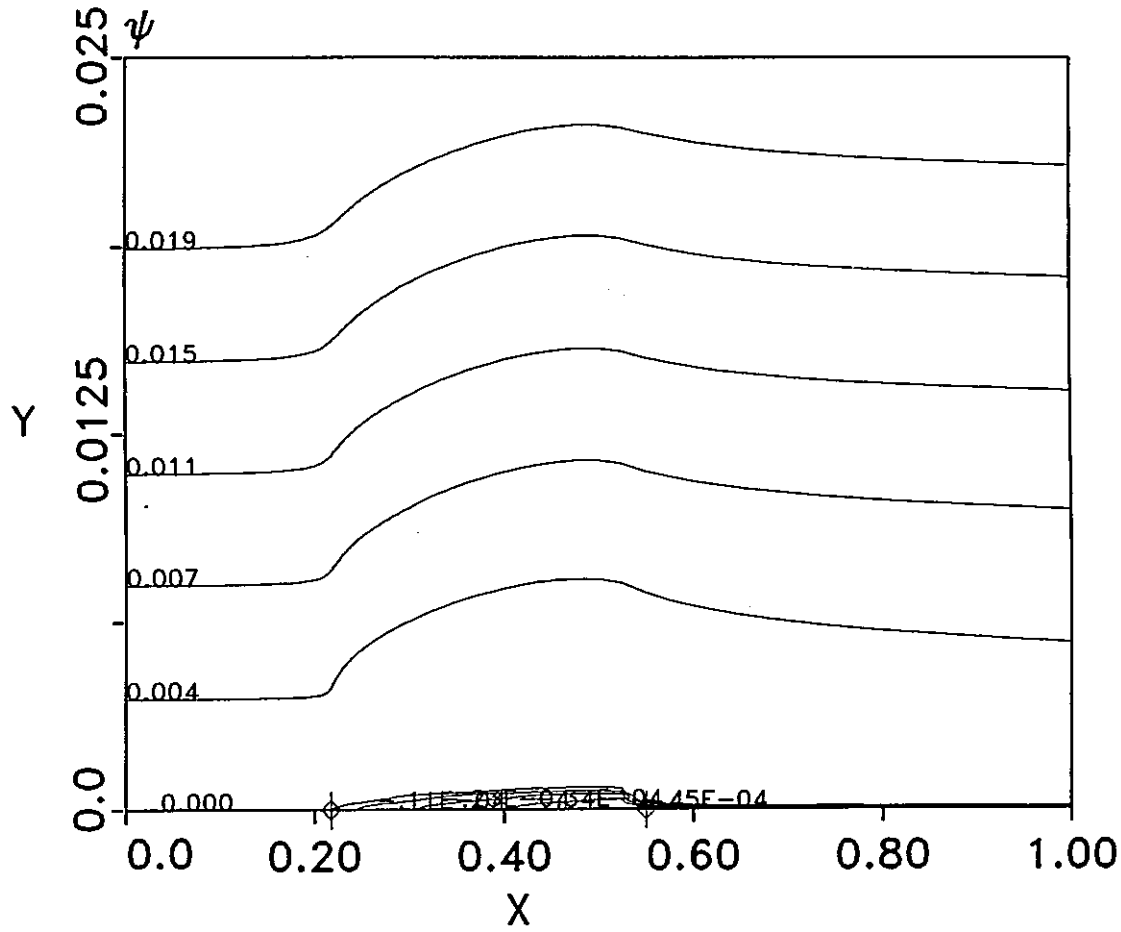


Fig 32 : Contours of dimensionless Stream Function for U_∞ = 0.10 m/s, T_∞ = 21.15 ^{o}C and a 71 by 38 non-uniform grid

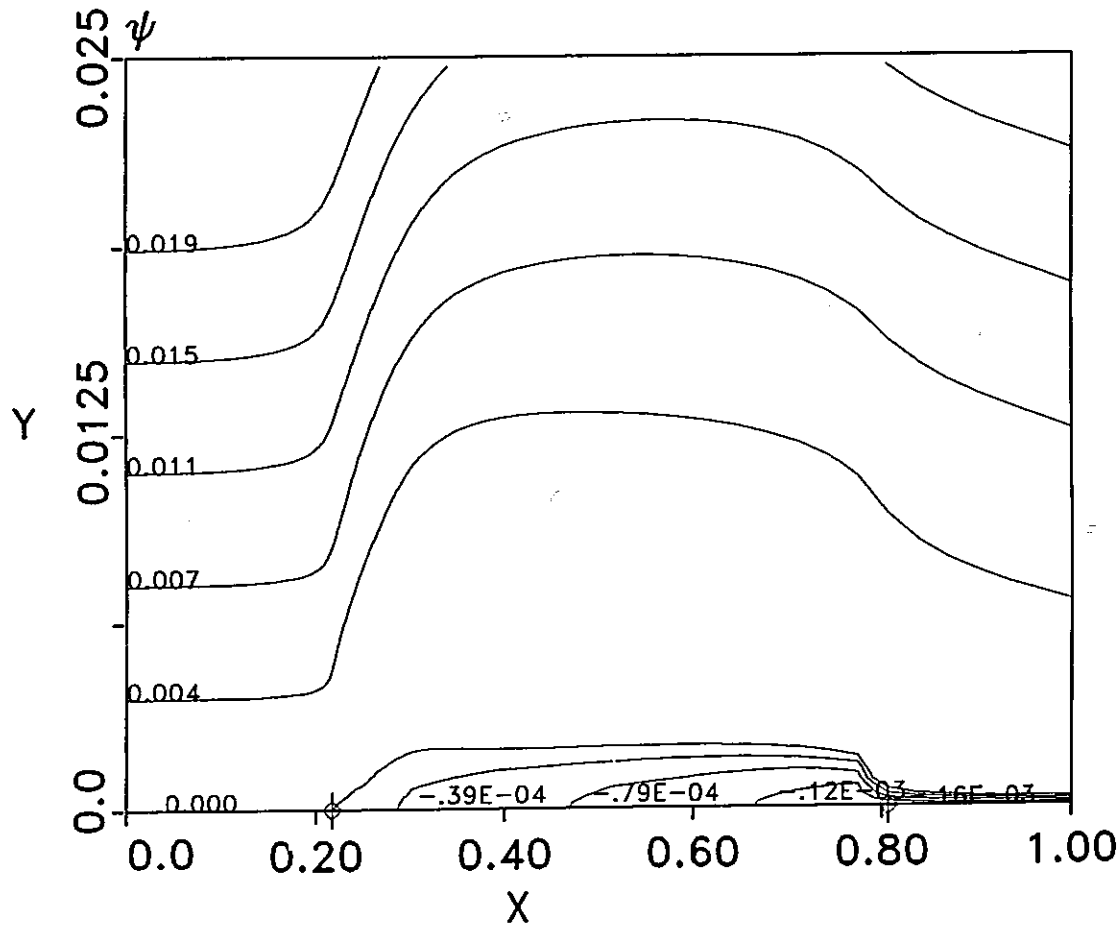


Fig 33 : Contours of dimensionless Stream Function for U_∞ = 0.10 m/s, T_∞ = 21.15 ^{o}C and a 67 by 36 non-uniform grid, for increased ice sheet length

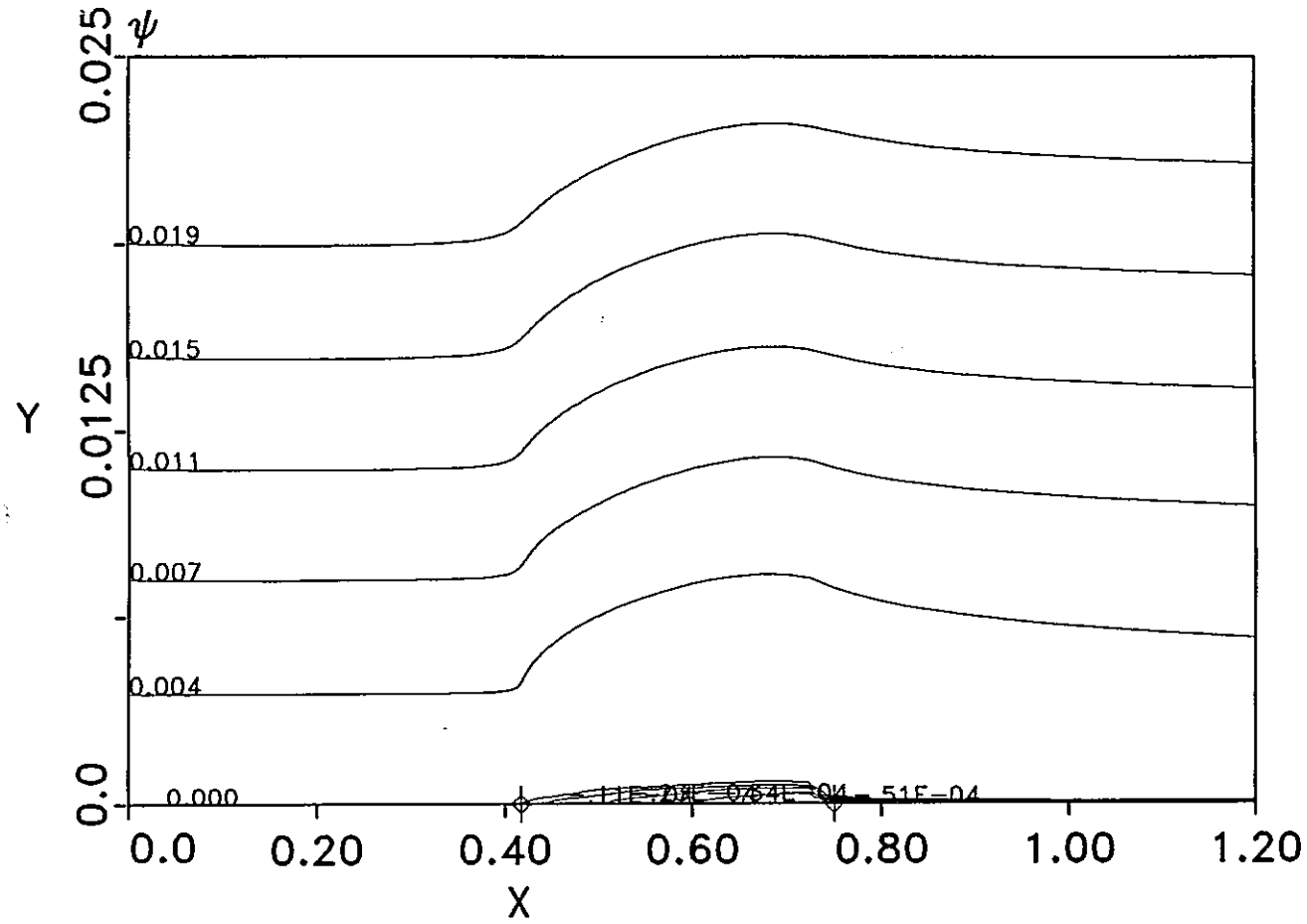


Fig 34 : Contours of dimensionless Stream Function for U_∞ = 0.10 m/s, T_∞ = 21.15 ^{0}C and a 76 by 38 non-uniform grid, for increased lead domain

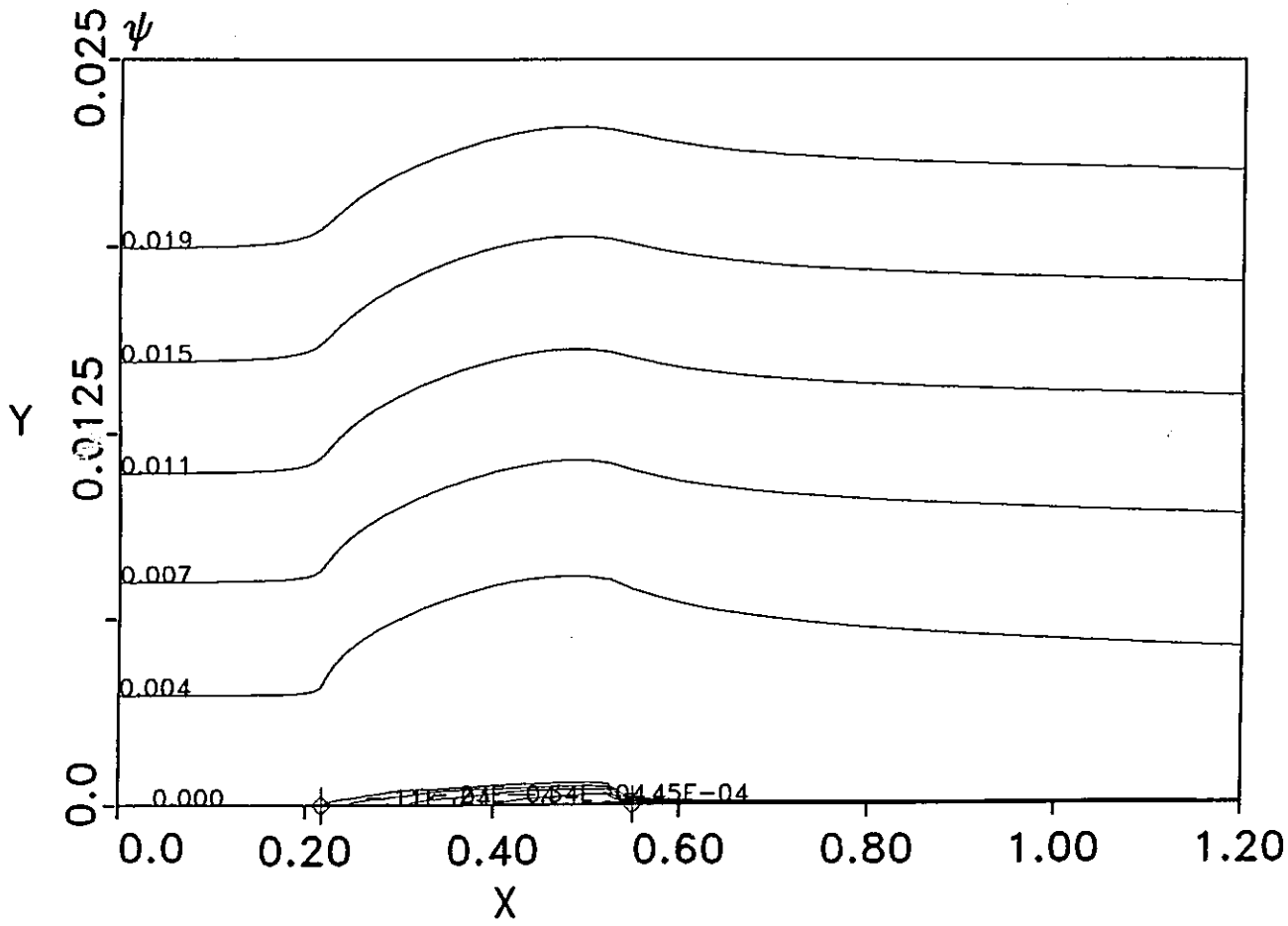


Fig 35 : Contours of dimensionless Stream Function for U_∞ = 0.10 m/s, T_∞ = 21.15 ^{o}C and a 77 by 38 non-uniform grid, for increased trail domain

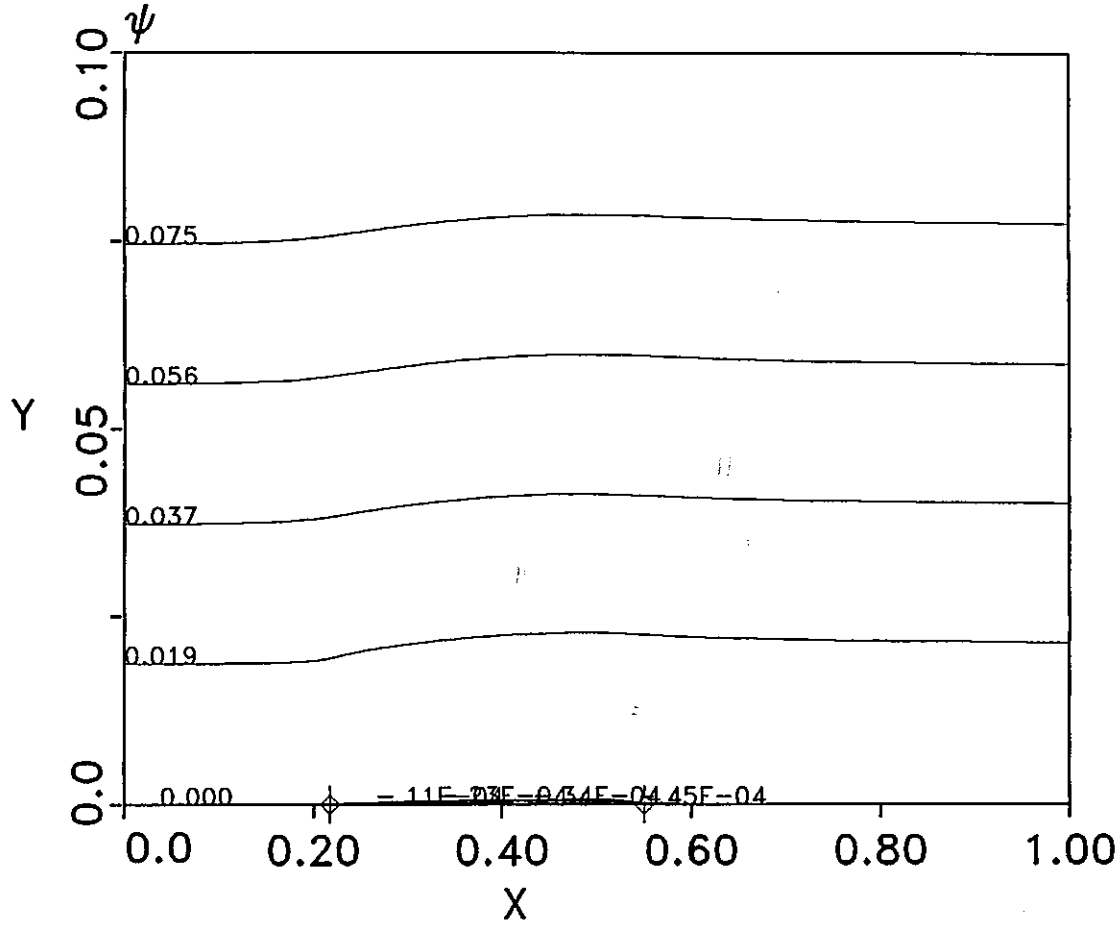


Fig 36 : Contours of dimensionless Stream Function for U_∞ = 0.10 m/s, T_∞ = 21.15 ^{o}C and a 71 by 38 non-uniform grid, for complete y domain

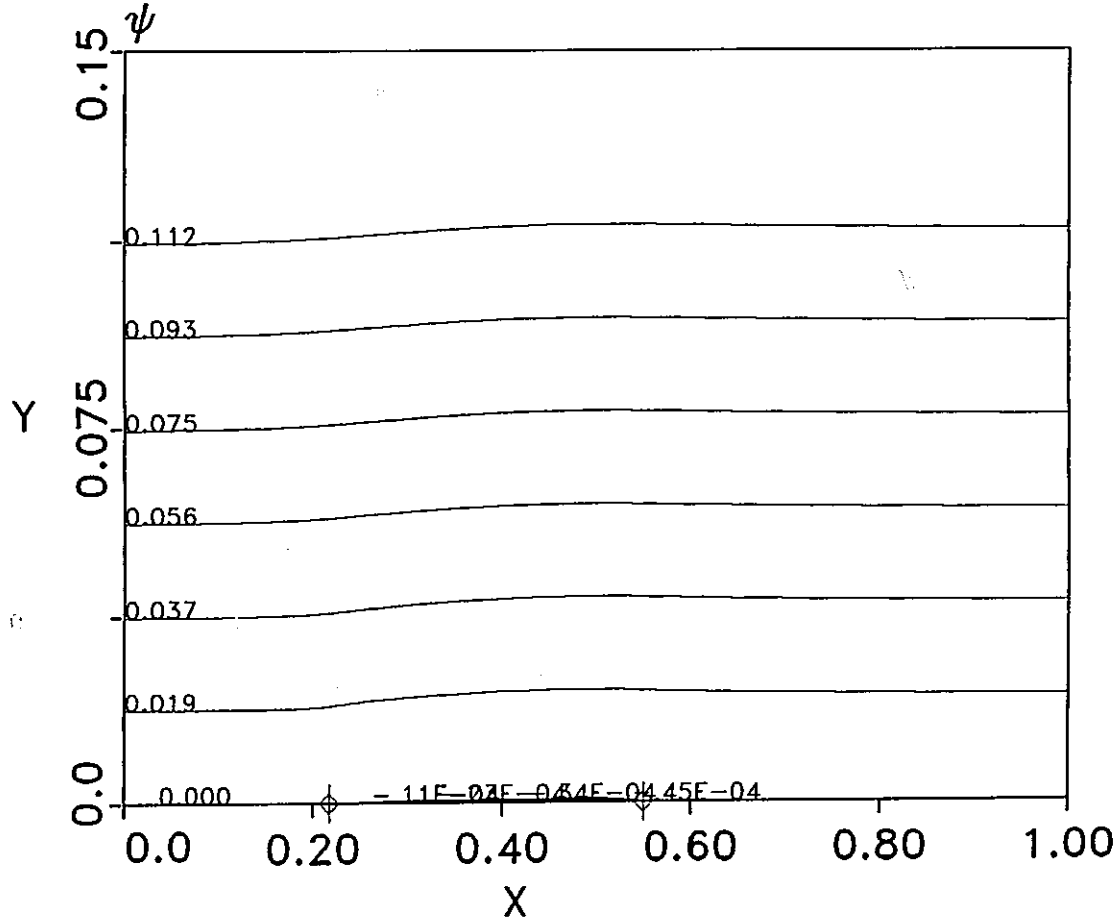


Fig 37 : Contours of dimensionless Stream Function for U_∞ = 0.10 m/s, T_∞ = 21.15 ^{0}C and a 71 by 41 non-uniform grid, for increased y domain

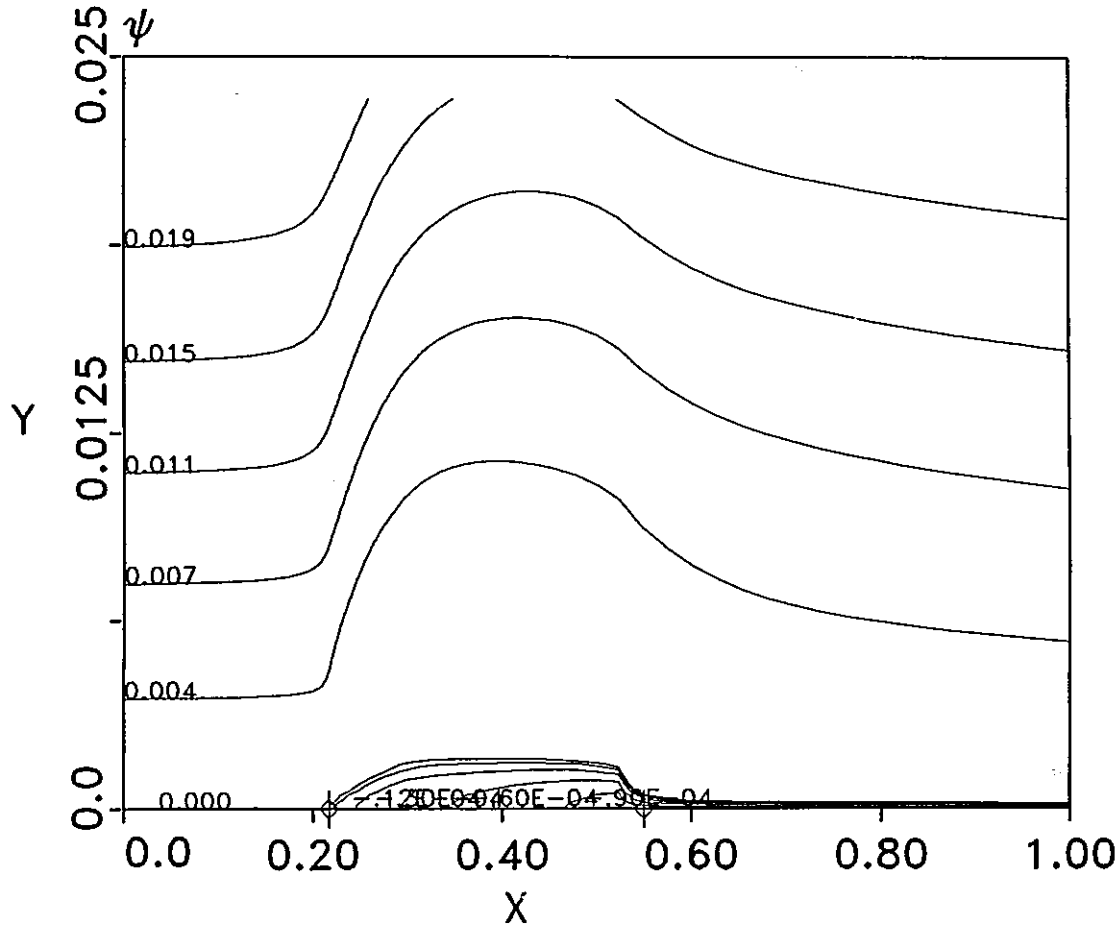


Fig 38 : Contours of dimensionless Stream Function for U_∞ = 0.02 m/s, T_∞ = 21.15 ^{o}C and a 71 by 38 non-uniform grid

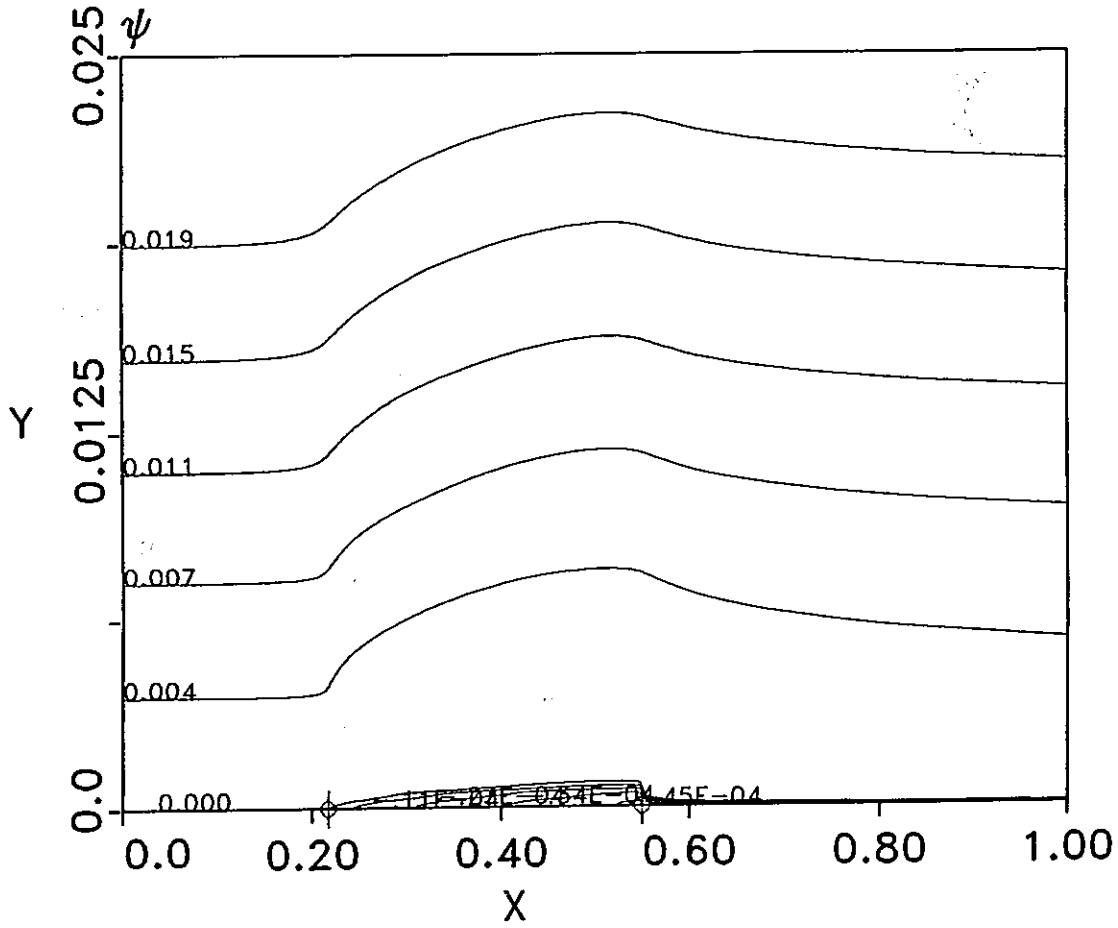


Fig 39 : Contours of dimensionless Stream Function for U_∞ = 0.10 m/s, T_∞ = 21.15 ^{o}C and a 96 by 38 non-uniform grid

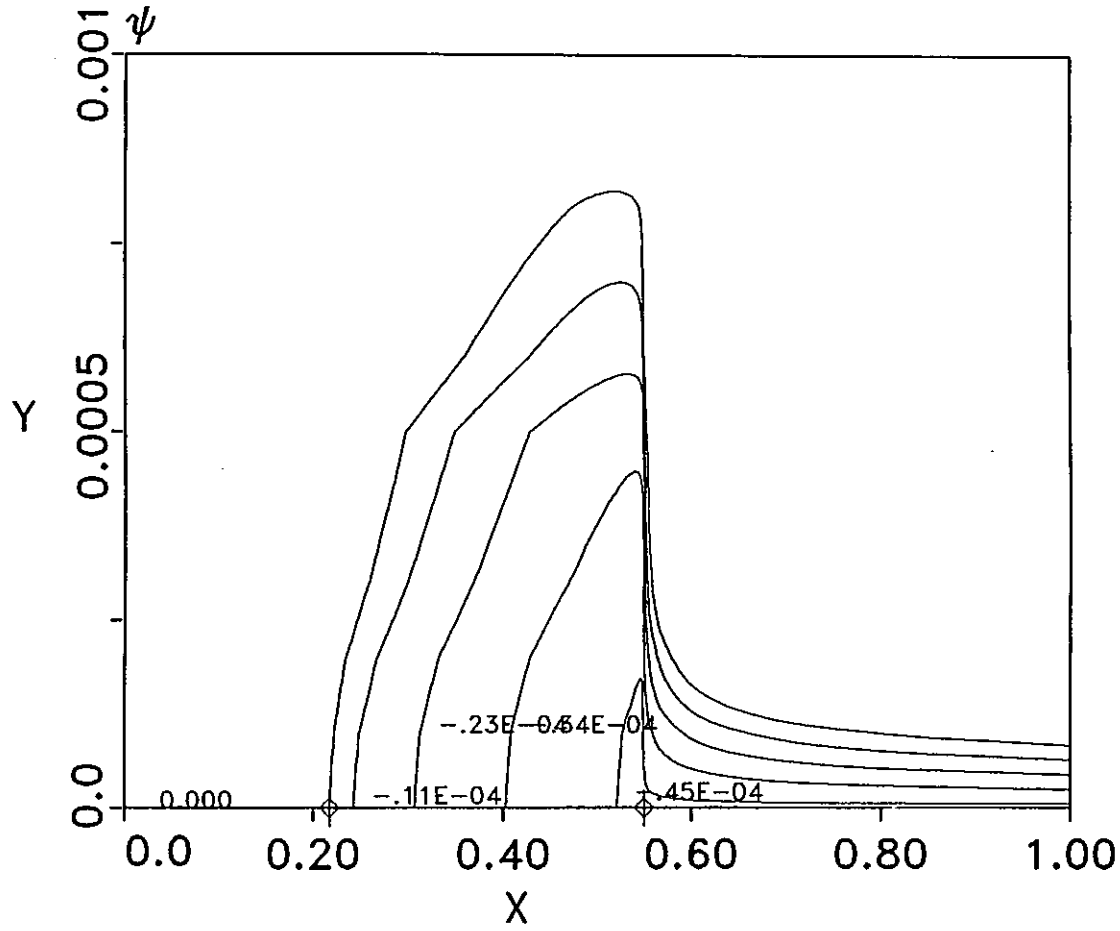


Fig 40 : Contours of dimensionless Stream Function for U_∞ = 0.10 m/s, T_∞ = 21.15 ^{o}C and a 96 by 38 non-uniform grid

```
      program ice_flom
*************************************************************************
*
*************************************************************************
      include 'paramifm.for'
      DOUBLE PRECISION  f,w,fi,s,azmf,amall,
     *  ro,u,v,f1,w1,delt,deltl,r,azmw,azmfi,
     *  fil,f2,w2,fi2,ul,azwl,azfil,azfl,
     *  vl,u2,v2,a,b,c,azf,azm,ra,al,am,
     *  d,e,ft,wt,fit,ftt,dfidy,dfidys,
     *  dfidx,dfidxs,dwdy,dwdys,v7,v4,v5,v3,aad,
     *  dwdx,dwdxs,dfdysq,dfdxsq,time,x,y,
     *   re,enfhx,enfhy,fmax,q,roref,
     *  romax,znuref,diffu,alphl,tmax,alph,chrl,uinf,chng,grav,
     *   thecon, fusion,dfidyw,trap,vf,vi,xf,xi,xdel,wp,wpp,xddel,azmc
c     double precision x,y
      include  'commbal.for'
c      write(*,*)alphl,tmax,znuref,roref,romax,diffu
       diffu = 1.3083d-07
       znuref =  1.7533d-06
       roref = 999.84d0
       grav = 9.81
       romax =  999.972d0
       alphl =  9.297173d-06
       tmax =  4.029325d0
       pr = 13.0d0
       thecon = 0.5687d0
        fusion = 334944.d0
      alph = 1.351261499d-07
      open(unit=1,status = 'old',file = 'icc_1x67.dat')
      do 1000 i = 1,in
      read(1,*) x(i)
1000  continue
      close(unit = 1)
      open(unit=1,status = 'old',file = 'ice_ym.dat')
      do 1001 i = 1,jn
      read(1,*) y(i)
1001  continue
      close(unit = 1)
      write(*,*) 'enter ice start node'
      read(*,15) ki
      write(*,*) ki
      write(*,*) 'enter ice end node'
      read(*,15) ke
      write(*,*) ke
      call strfife
      q = 1.894816d0
c     write(*,*) 'enter rayleigh number'
c     read(*,*) ra
c     write(*,*) ' enter characteristic length '
c      read(*,*) chrl
      write(*,*) ' enter uinf'
       read(*,*) uinf
```

```
c         write(*,*) chrl,uinf
        chrl = 1.d0
        write(*,*) 'enter r'
        read(*,*) r
        tdelt = dabs(tmax/r)
         ra = grav*alphl*romax*tdelt**q*chrl**3/(roref*znuref*alph)
         re = uinf*chrl/znuref
c       write(*,*) 'enter  reynolds number'
c       read(*,*) re
        write(*,*) ra,r,re
        write(*,*) 'enter nmax'
        read(*,15)nmax
15      format(i8)
        write(*,*) nmax
        write(*,*) 'enter convergence criterion'
        read(*,16) cc
16      format(f10.0)
        write(*,*) 'enter time step'
        read(*,16) delt
        write(*,*) 'enter file step'
        read(*,15) krep
        write(*,*) krep
C       znuref = znuref/(uinf*chrl)
c**************************************************************************
c   upstream stream function initialisation
c**************************************************************************
c********666 and 667 are to be commented out if data is read in****
c       f(1,1) = 0.d0
c       do 666 j = 1,jnm
c       write(*,*) f(1,j)
c       f(1,j+1) = f(1,j) + 1.0d0 * (y(j+1) - y(j))
c666    continue
c       do 667 j = 1,jn
c        do 667 i = 1,in
c           f(i,j) = f(1,j)
c667     continue
        xdde1 = ( x(in) - x(in-2))/(x(in-1)-x(in-2))
20      continue
        time=time+delt
        k=k+1
        if(mod(k,krep).eq.0) then
          open(unit=1,status='new',form='unformatted'
     *         ,file='ice_flom.dat')
           write(1)in,jn,time
            write(1) x,y,f,w,fi,ro,u,v
              write(1) delt,chrl,r,uinf
                close(unit=1)
        endif
        do 91 i = 1,in
          do 91 j=1,jn
          w1(i,j) = w(i,j)
          f1(i,j) = f(i,j)
          fi1(i,j) = fi(i,j)
```

```
91        continue
          do 11 j = 3,jn-2
            do 11 i = 2,in-1
          u(i,j)=(8.*f(i,j+1)-f(i,j+2)-8.*f(i,j-1)+f(i,j-2))/
     *   (8.*y(j+1)-8.*y(j-1)-y(j+2)+y(j-2))
11        continue
          do 684 i = 3,in-2
            do 684 j = 2,jnm
          v(i,j)=(-8.*f(i+1,j)+f(i+2,j)+8.*f(i-1,j)-f(i-2,j))/
     *   (8.*x(i+1)-8.*x(i-1)-x(i+2)+x(i-2))
684       continue
          do 1111 i = 2,inm
           u(i,2) = (-3.* f(i,2)+6.*f(i,3)-f(i,4)-2.*f(i,1))/
     *    (5.*(y(3)-y(2))-(y(4)-y(3))+2.*(y(2)-y(1)))
           u(i,jnm)= -(-f(i,jnm-2)+6.*f(i,jnm-1)-3.*f(i,jnm)
     *    -2.*f(i,jn))/
     *    (5.*(y(jnm)-y(jn-2))-(y(jn-2)-y(jn-3))+
     *    2.*(y(jn)-y(jnm)))
1111      continue
            do 1112 j = 2,jnm
           v(2,j) =(3.*f(2,j)-6.*f(3,j)+f(4,j)+2.*f(1,j))/
     *    (5.*(x(3)-x(2))-(x(4)-x(3))+2.*(x(2)-x(1)))
           v(inm,j)= -(f(inm-2,j)-6.*f(inm-1,j)+3.*f(inm,j)
     *    +2.*f(in,j))/
     *    (5.*(x(inm)-x(in-2))-(x(in-2)-x(in-3))+2.*(x(in)-x(inm)))
1112       continue
          alph = 1.d0/(pr*re)
          enfhx = 0.
          enfhy = 0.
          dfidy = 0.
          dfidys = 0.
          dwdy = 0.
          dwdys = 0.
          dfdy = 0.
          dfdysq = 0.
C*************** ENERGY FIRST HALF ***********************************
          do 18 j = 2,jnm
          do 53 i = 2,inm
          enfhy = 0.
          enfhx = 0.
          dfidy = 0.
          dfidys = 0.
          enfhy = ((y(j+1)-y(j))*(y(j)-y(j-1))*(y(j+1)-y(j-1)))
          enfhx = ((x(i+1)-x(i))*(x(i)-x(i-1))*(x(i+1)-x(i-1)))
c         dfidy = (fi(i,j+1)*(y(j)-y(j-1))**2-fi(i,j-1)*
c     *       (y(j+1)-y(j))**2
c     *      - fi(i,j)*((y(j)-y(j-1))**2- (y(j+1)-y(j))**2))
c     *  /enfhy
          dfidys = 2.*(fi(i,j+1)*(y(j)-y(j-1))+fi(i,j-1)*
     *     (y(j+1)-y(j))
     *    -fi(i,j)*(y(j+1)-y(j-1)))/enfhy
c         a(i) =(1./enfhx)*(delt*(x(i)-x(i-1))*alph-u(i,j)*delt*
c     * (x(i)-x(i-1))**2/2.)
```

```
c       b(i)=1.d0-(u(i,j)*delt/(2.*enfhx))*((x(i)-x(i-1))**2
c     *     -(x(i+1)-x(i))**2)
c     *  + (delt*alph)*(x(i+1)-x(i-1))/enfhx
c       c(i)=(1./enfhx)*(u(i,j)*delt*(x(i+1)-x(i))**2/2.+
c        delt*(x(i+1)-x(i))*alph)
c             d(i) =(-v(i,j)*dfidy+dfidys*alph)*delt/2.
c     *                + fi(i,j)
        if(u(i,j).le.0.d0.and.v(i,j).le.0.d0)then
           dfidy = (fi(i,j+1)-fi(i,j))/(y(j+1)-y(j))
            go to 1008
         elseif(u(i,j).le.0.d0.and.v(i,j).gt.0.d0) then
           dfidy = (fi(i,j)-fi(i,j-1))/(y(j)-y(j-1))
            go to 1008
        else
           go to 1108
        endif
1008       a(i) =(1./enfhx)*(delt*(x(i)-x(i-1))*alph)-u(i,j)*delt/
     * ((x(i+1)-x(i))*2.)
           b(i)=1.d0 - u(i,j)*delt/(2.*(x(i+1)-x(i)))
     *  + (delt*alph)*(x(i+1)-x(i-1))/enfhx
           c(i)=(1./enfhx)*(delt*(x(i+1)-x(i))
     *   *alph)
             d(i) =(-v(i,j)*dfidy+dfidys*alph)*delt/2.
     *                + fi(i,j)
         go to 53
1108       if(u(i,j).gt.0.d0.and.v(i,j).le.0.d0) then
              dfidy = (fi(i,j+1)-fi(i,j))/(y(j+1)-y(j))
               go to 503
          elseif(u(i,j).gt.0.d0.and.v(i,j).gt.0.d0) then
              dfidy = (fi(i,j)-fi(i,j-1))/(y(j)-y(j-1))
              go to 503
        else
             go to 53
           endif
503           a(i) =(1./enfhx)*(delt*(x(i)-x(i-1))*alph)
             b(i)=1.d0 + u(i,j)*delt/(2.*(x(i)-x(i-1)))
     *  + (delt*alph)*(x(i+1)-x(i-1))/enfhx
             c(i)=(1./enfhx)*(delt*(x(i+1)-x(i))*alph)+
     *   u(i,j)*delt/(2.*(x(i)-x(i-1)))
             d(i) =(-v(i,j)*dfidy+dfidys*alph)*delt/2.
     *                + fi(i,j)
53      continue
        ll = 1
c        lm = 2
         lm = 1
         al = 1.d0
c        am = 0.d0
         am = fi(in-2,j)+xddel*(fi(in-1,j)-fi(in-2,j))
         call gtriv(a,b,c,d,in,e,ft,fit,ll,al,lm,am)
       do 19 kk = 1,in
        fi2(kk,j) = fit(kk)
19      continue
18      continue
```

```
c       enfr = enfm1/enfm2
c       write(*,*)' first half energy ratio:', enfr
        do 801 i = 1,in
        if(i.lt.ki) then
          fi2(i,1) = fi2(i,2)
           go to 8112
        elseif(i.ge.ki.and.i.le.ke)then
          fi2(i,1) = 0.0d0
          go to 8112
        else
          fi2(i,1) = fi2(i,2)
        endif
C****** set gradient of temp. as zero at the leading and trailing edge
8112    fi2(i,jn) = 1.0d0
801     continue
        do 811 i = 1,in
        a(i) = 0.
        b(i) = 0.
        c(i) = 0.
        d(i) = 0.
811     continue
C***************** ENERGY SECOND HALF ************************
         do 22 i  = 2,inm
         do 54 j = 2,jnm
        enfhy = 0.
        enfhx = 0.
        dfidx = 0.
        dfidxs = 0.
        enfhy = ((y(j+1)-y(j))*(y(j)-y(j-1))*(y(j+1)-y(j-1)))
        enfhx = ((x(i+1)-x(i))*(x(i)-x(i-1))*(x(i+1)-x(i-1)))
c       dfidx = (fi2(i+1,j)*(x(i)-x(i-1))**2-fi2(i-1,j)*
c     *    (x(i+1)-x(i))**2
c       *          -fi2(i,j)*((x(i)-x(i-1))**2-(x(i+1)-x(i))**2))/enfhx
        dfidxs = 2.*(fi2(i+1,j)*(x(i)-x(i-1))+fi2(i-1,j)*
     *      (x(i+1)-x(i))
     *    -fi2(i,j)*(x(i+1)-x(i-1)))/enfhx
c       a(j)=(1./enfhy)*(delt*(y(j)-y(j-1))*alph-v(i,j)*delt*
c     * (y(j)-y(j-1))**2/2.)
c       b(j)=1.d0-(v(i,j)*delt/(2.*enfhy))*((y(j)-y(j-1))**2
c     *    -(y(j+1)-y(j))**2)
c     *  + (delt*alph)*(y(j+1)-y(j-1))/enfhy
c       c(j)=(1./enfhy)*(v(i,j)*delt*(y(j+1)-y(j))**2/2.+
c     * delt*(y(j+1)-y(j))*alph)
c             d(j) =(-u(i,j)*dfidx+dfidxs*alph)*delt/2.
c     *                  + fi2(i,j)
        if(v(i,j).le.0.d0.and.u(i,j).le.0.d0) then
             dfidx = (fi2(i+1,j)-fi2(i,j))/(x(i+1)-x(i))
              go to 1009
         elseif(v(i,j).le.0.d0.and.u(i,j).gt.0.d0) then
            dfidx = (fi2(i,j)-fi2(i-1,j))/(x(i)-x(i-1))
              go to 1009
        else
               go to 1109
```

```
        endif
1009       a(j) =(1./enfhy)*(delt*(y(j)-y(j-1))*alph)-v(i,j)*delt/
     * ((y(j+1)-y(j))*2.)
        b(j)=1.d0 - v(i,j)*delt/(2.*(y(j+1)-y(j)))
     *  + (delt*alph)*(y(j+1)-y(j-1))/enfhy
        c(j)=(1./enfhy)*(delt*(y(j+1)-y(j))
     *   *alph)
             d(j) =(-u(i,j)*dfidx+dfidxs*alph)*delt/2.
     *                + fi2(i,j)
          go to 54
1109    if(v(i,j).gt.0.d0.and.u(i,j).le.0.d0) then
            dfidx = (fi2(i+1,j)-fi2(i,j))/(x(i+1)-x(i))
            go to 504
         elseif(v(i,j).gt.0.d0.and.u(i,j).gt.0.d0) then
           dfidx = (fi2(i,j)-fi2(i-1,j))/(x(i)-x(i-1))
           go to 504
        else
           go to 54
         endif
504          a(j) =(1./enfhy)*(delt*(y(j)-y(j-1))*alph)
            b(j)=1.d0 + v(i,j)*delt/(2.*(y(j)-y(j-1)))
     *  + (delt*alph)*(y(j+1)-y(j-1))/enfhy
             c(j)=(1./enfhy)*(delt*(y(j+1)-y(j))*alph)+
     *   v(i,j)*delt/(2.*(y(j)-y(j-1)))
             d(j) =(-u(i,j)*dfidx+dfidxs*alph)*delt/2.
     *                + fi2(i,j)
c    ens1=(v(i,j)*delt/(2.*enfhy))*((y(j)-y(j-1))**2-(y(j+1)-y(j))**2)
c         ens2=(delt/pr)*(y(j+1)-y(j-1))/enfhy
c        ensm1 = dmax1(ens1,ensm1)
c        ensm2 = dmax1(ens2,ensm2)
54        continue
        if(i.lt.ki) then
            ll = 2
            lm = 1
            al = 0.d0
            am = 1.0d0
          go to 545
        elseif(i.ge.ki.and.i.le.ke) then
            ll = 1
            lm = 1
            al = 0.d0
            am = 1.0d0
          go to 545
        else
            ll = 2
            lm = 1
            al = 0.d0
            am = 1.d0
        endif
545      call gtriv(a,b,c,d,jn,e,ft,fit,ll,al,lm,am)
        do 23 kk = 1,jn
        fi(i,kk) = fit(kk)
23      continue
```

```
22       continue
c       ensr = ensm1/ensm2
c       write(*,*)' second half energy ratio:',ensr
        do 802 j = 1,jn
        fi(1,j) = 1.d0
c       fi(in,j) = f(inm,j)
        fi(in,j) = fi(in-2,j)+xddel*(fi(in-1,j)-fi(in-2,j))
802     continue
        fi(1,1) = 0.d0
        do 812 j = 1,jn
        a(j) = 0.
        b(j) = 0.
        c(j) = 0.
        d(j) = 0.
812     continue
C******************VORTICITY FIRST HALF ******************************
        xdel = x(in) - x(inm)
        do 1120 j = 2,jn
c*************************************************************************
c       Two options: either set a vorticity or assume zero vorticity
c*************************************************************************
        w(1,j) = 0.d0
c       w(1,j) = (-8.*f(2,j)+f(3,j)+7.*f(1,j))/(
c     *    3.5*(x(2)-x(1))**2-(x(2)-x(1))*
c     *   (x(3)-x(2))-(x(3)-x(2))**2/2.)
c*************************************************************************
c Two options; either gradient = 0 or w(in,j) = 0.trying out the
c former
c*************************************************************************
        w(in,j) = w(in-2,j)+xddel*(w(in-1,j)-w(in-2,j))
c       w(in,j) = w(inm,j)
c         w(in,j) = 0.d0
c************** shapiro and o'brien  ********************************
c        wp = w(in,j)+(u(inm,j)*delt*(w(inm,j)-w(in,j))/xdel)
c        wpp = w(in,j-1)+(u(inm,j)*delt*
c     *           (w(inm,j-1)-w(in,j-1))/xdel)
c        w(in,j) = wp + (v(inm,j)*delt*(wpp-wp)/xdel)
1120    continue
        do 12 j = 2,jnm
        do 51 i = 2,inm
        enfhy = 0.
        enfhx = 0.
        dwdy = 0.
        dwdys = 0.
        dfidx = 0.
        enfhy = ((y(j+1)-y(j))*(y(j)-y(j-1))*(y(j+1)-y(j-1)))
        enfhx = ((x(i+1)-x(i))*(x(i)-x(i-1))*(x(i+1)-x(i-1)))
c       dwdy = (w(i,j+1)*(y(j)-y(j-1))**2-w(i,j-1)*
c     *        (y(j+1)-y(j))**2
c       *           -w(i,j)*((y(j)-y(j-1))**2-(y(j+1)-y(j))**2))/enfhy
        dwdys = 2.*(w(i,j+1)*(y(j)-y(j-1))+w(i,j-1)*(y(j+1)-y(j))
     *    -w(i,j)*(y(j+1)-y(j-1)))/enfhy
        v4 = fi(i,j)- r
```

```
        v5 = q - 2.d0
        v3 = dabs(v4)
        tdelt = dabs(tmax/r)
c       chng = chrl/uinf**2*q*9.81*alphl*romax*tdelt**q/roref
        chng = q*ra/(re**2*pr)
        if(v4.eq.0.)then
          aad = 0.d0
           go to 601
        else
           aad =chng*v3**v5*(fi(i,j)-r)
        endif
601     dfidx = (fi(i+1,j)*(x(i)-x(i-1))**2-fi(i-1,j)*
     *       (x(i+1)-x(i))**2
     *    -fi(i,j)*((x(i)-x(i-1))**2-(x(i+1)-x(i))**2))/enfhx
c       a(i) =(1./enfhx)*((1.d0/re)*delt*(x(i)-x(i-1))-
c     *   u(i,j)*delt* (x(i)-x(i-1))**2/2.)
c       b(i)=1.d0-(u(i,j)*delt/(2.*enfhx))*((x(i)-x(i-1))**2
c     *    -(x(i+1)-x(i))**2)
c     *  + (delt*(1.d0/re))*(x(i+1)-x(i-1))/enfhx
c       c(i)=(1./enfhx)*(u(i,j)*delt*(x(i+1)-x(i))**2/2.+
c     * delt*(1.d0/re)*(x(i+1)-x(i)))
c             d(i) =(-v(i,j)*dwdy+(1.d0/re)*dwdys)*delt/2.
c     *                + w(i,j)+aad*delt/2.*dfidx
        if(u(i,j).le.0.d0.and.v(i,j).le.0.d0) then
           dwdy = (w(i,j+1)-w(i,j))/(y(j+1)-y(j))
            go to 1006
        elseif(u(i,j).le.0.d0.and.v(i,j).gt.0.d0) then
           dwdy = (w(i,j)-w(i,j-1))/(y(j)-y(j-1))
           go to 1006
        else
          go to 1106
        endif
1006        a(i) =(1./enfhx)*((1.d0/re)*delt*(x(i)-x(i-1)))
     *   -u(i,j)*delt
     * /((x(i+1)-x(i))*2.)
        b(i)=1.d0- u(i,j)*delt/(2.*(x(i+1)-x(i)))
     *  + (delt*(1.d0/re))*(x(i+1)-x(i-1))/enfhx
        c(i)=(1./enfhx)*delt*(1.d0/re)*(x(i+1)-x(i))
             d(i) =(-v(i,j)*dwdy+(1.d0/re)*dwdys)*delt/2.
     *                + w(i,j)+aad*delt/2.*dfidx
         go to 51
c   vof1=(u(i,j)*delt/(2.*enfhx))*((x(i)-x(i-1))**2-(x(i+1)-x(i))**2)
c       vof2= (delt)*(x(i+1)-x(i-1))/enfhx
c       vofm1 = dmax1(vof1,vofm1)
c       vofm2 = dmax1(vof2,vofm2)
1106    if(u(i,j).gt.0.d0.and.v(i,j).le.0.d0) then
             dwdy = (w(i,j+1)-w(i,j))/(y(j+1)-y(j))
             go to 501
         elseif(u(i,j).gt.0.d0.and.v(i,j).gt.0.d0) then
             dwdy = (w(i,j)-w(i,j-1))/(y(j)-y(j-1))
             go to 501
c
        else
```

```
          go to 51
         endif
501      a(i) =(1./enfhx)*((1.d0/re)*delt*(x(i)-x(i-1)))
        b(i)=1.d0+ u(i,j)*delt/(2.*(x(i)-x(i-1)))
     *  + (delt*(1.d0/re))*(x(i+1)-x(i-1))/enfhx
        c(i)=(1./enfhx)*delt*(1.d0/re)*(x(i+1)-x(i))+
     *    u(i,j)*delt/(2.*(x(i)-x(i-1)))
             d(i) =(-v(i,j)*dwdy+(1.d0/re)*dwdys)*delt/2.
     *                + w(i,j)+aad*delt/2.*dfidx
51       continue
         l1 = 1
c************ shapiro and o'brien ***********************************
         lm = 1
c        lm = 2
c******** zero vort tryout****
c         lm = 1
c        al = (-8.*f(2,j)+f(3,j)+7.*f(1,j))/(3.5*(x(2)-x(1))
c     *     **2-(x(2)-x(1))*
c     *   (x(3)-x(2))-(x(3)-x(2))**2/2.)
         al = 0.d0
         am = w(in-2,j)+xddel*(w(in-1,j)-w(in-2,j))
c        am = 0.d0
c******************shapiro and o'brien ***************************** c
am = w(in,j)
         call gtriv(a,b,c,d,in,e,ft,wt,l1,al,lm,am)
         do 13 kk = 1,in
          w2(kk,j) = wt(kk)
c        write(*,*) w(KK,j)
13       continue
12        continue
c        do 8804 j = 2,jnm
c        do 8804 i = 2,inm
         do 8804 i = ki,ke
         dfidyl = (8.*fi(i,2)-fi(i,3)-7.*fi(i,1))/
     *    (7.*(y(2)-y(1))-(y(3)-y(2)))
c        v(i,1) = dfidyl * tdelt*thecon/(uinf*roref*fusion*chrl)
         v(i,1) = dfidyl * tdelt*thecon/(znuref*roref*fusion*re)
8804     continue
         do 803 i = 2,inm
c************ 2nd order vort*********************************
         if(i.lt.ki ) then
            w2(i,1) = 0.d0
             go to 8113
          elseif(i.ge.ki.and.i.le.ke) then
         w2(i,1)=(-8.*f(i,2)+f(i,3)+7.*f(i,1))/(3.5*(y(2)-y(1))**2
     *   -(y(2)-y(1))*(y(3)-y(2))-(y(3)-y(2))**2/2.)
     * - 2.*(f(i+1,1)*(x(i)-x(i-1))+f(i-1,1)*(x(i+1)-x(i))
     *    -f(i,1)*(x(i+1)-x(i-1)))/
     *    ((x(i+1)-x(i))*(x(i)-x(i-1))*(x(i+1)-x(i-1)))
          go to 8113
         else
             w2(i,1) = 0.d0
         endif
```

```
c       w2(i,jn)=(-8.*f(i,jnm)+f(i,jn-2)+7.*1.0*(y(jnm)-y(jn))-
c     *  1.0*(y(jn-2)-y(jnm)))/
c     *         (3.5*(y(jnm)-y(jn))**2-(y(jnm)-y(jn))
c     *  * (y(jn-2)-y(jnm))-(y(jn-2)-y(jnm))**2/2.)
c*************************************************************************
c   setting vorticity on the upper surface equal to zero
c*************************************************************************
8113    w2(i,jn) = 0.d0
c*********** freeing the lid ***************************************
c        w2(i,jn) = w2(i,jnm)
c*************************************************************************
c    newmann upper surface vort. tryout
c*************************************************************************
c        w2(i,jn) = w2(i,jnm)
803     continue
c       vofr = vofm1/vofm2
c       write(*,*)' first half vort :', vofr
        do 813 i = 1,in
        a(i) = 0.
        b(i) = 0.
        c(i) = 0.
        d(i) = 0.
813     continue
C***************** VORTICITY SECOND HALF ***********************
        do 1121 i = 1,in
c       w2(i,1)=(-8.*f(i,2)+f(i,3))/(3.5*(y(2)-y(1))**2
c     *  -(y(2)-y(1))*
c     *   (y(3)-y(2))-(y(3)-y(2))**2/2.)
c*********   2nd order vort***********
        if(i.lt.ki) then
            w2(i,1) = 0.d0
             go to 1221
        elseif(i.ge.ki.and.i.le.ke) then
        w2(i,1) = (-8.*f(i,2)+f(i,3)+7.*f(i,1))/
     *    (3.5*(y(2)-y(1))**2
     *  -(y(2)-y(1))*(y(3)-y(2))-(y(3)-y(2))**2/2.)
     * - 2.*(f(i+1,1)*(x(i)-x(i-1))+f(i-1,1)*(x(i+1)-x(i))
     *    -f(i,1)*(x(i+1)-x(i-1)))/
     *    ((x(i+1)-x(i))*(x(i)-x(i-1))*(x(i+1)-x(i-1)))
         go to 1221
        else
           w2(i,1)= 0.d0
        endif
c       w2(i,jn)=(-8.*f(i,jnm)+f(i,jn-2)+7.*1.0*(y(jnm)-y(jn))-
c     *  1.0*(y(jn-2)-y(jnm)))/
c     *         (3.5*(y(jnm)-y(jn))**2-(y(jnm)-y(jn))
c     *  * (y(jn-2)-y(jnm))-(y(jn-2)-y(jnm))**2/2.)
c*********** upper surface vort set zero*******************
1221    w2(i,jn) = 0.d0
c************ newmann upper vort*********************************
c         w2(i,jn) = w2(i,jnm)
c******** freeing the lid************************************
c          w2(i,jn) = w2(i,jnm)
```

```
1121    continue
        do 14 i  = 2,inm
        do 52 j = 2,jnm
        enfhx = 0.
        enfhy = 0.
        dwdx = 0.
        dwdxs = 0.
        dfidx = 0.
        enfhy = ((y(j+1)-y(j))*(y(j)-y(j-1))*(y(j+1)-y(j-1)))
        enfhx = ((x(i+1)-x(i))*(x(i)-x(i-1))*(x(i+1)-x(i-1)))
c       dwdx = (w2(i+1,j)*(x(i)-x(i-1))**2-w2(i-1,j)*(
c     *    x(i+1)-x(i))**2
c       *          -w2(i,j)*((x(i)-x(i-1))**2-(x(i+1)-x(i))**2))/enfhx
        dwdxs = 2.*(w2(i+1,j)*(x(i)-x(i-1))+w2(i-1,j)*(x(i+1)-x(i))
     *    -w2(i,j)*(x(i+1)-x(i-1)))/enfhx
        v7 = dabs(fi(i,j)-r)
        if(v7.eq.0.)then
          aad = 0.d0
           go to 602
        else
          aad = chng*(dabs(fi(i,j)-r))**(q-2.d0)*(fi(i,j)-r)
        endif
602     dfidx = (fi(i+1,j)*(x(i)-x(i-1))**2-fi(i-1,j)*
     *    (x(i+1)-x(i))**2
     *              -fi(i,j)*((x(i)-x(i-1))**2-(x(i+1)-x(i))**2))/enfhx
c       a(j)=(1./enfhy)*((1.d0/re)*delt*(y(j)-y(j-1))-v(i,j)*
c     *         delt*(y(j)-y(j-1))**2/2.)
c       b(j)=1.d0-(v(i,j)*delt/(2.*enfhy))*((y(j)-y(j-1))**2
c     *     -(y(j+1)-y(j))**2)
c     *  + (delt*(1.d0/re))*(y(j+1)-y(j-1))/enfhy
c       c(j)=(1./enfhy)*(v(i,j)*delt*(y(j+1)-y(j))**2/2.+
c     *  delt*(1.d0/re)*(y(j+1)-y(j)))
c            d(j) =(-u(i,j)*dwdx+(1.d0/re)*dwdxs)*delt/2.
c     *                + w2(i,j)+aad*delt/2.*dfidx
        if(v(i,j).le.0.d0.and.u(i,j).le.0.d0) then
            dwdx = (w2(i+1,j)-w2(i,j))/(x(i+1)-x(i))
             go to 1005
         elseif(v(i,j).le.0.d0.and.u(i,j).gt.0.d0) then
            dwdx = (w2(i,j)-w2(i-1,j))/(x(i)-x(i-1))
            go to 1005
        else
            go to 1105
        endif
1005       a(j) =(1./enfhy)*((1.d0/re)*delt*(y(j)-y(j-1)))-
     *     v(i,j)*delt
     * /((y(j+1)-y(j))*2.)
        b(j)=1.d0- v(i,j)*delt/(2.*(y(j+1)-y(j)))
     *  + (delt*(1.d0/re))*(y(j+1)-y(j-1))/enfhy
        c(j)=(1./enfhy)*delt*(1.d0/re)*(y(j+1)-y(j))
              d(j) =(-u(i,j)*dwdx+(1.d0/re)*dwdxs)*delt/2.
     *              + w2(i,j)+aad*delt/2.*dfidx
         go to 52
1105     if(v(i,j).gt.0.d0.and.u(i,j).le.0.d0) then
```

```
            dwdx = (w2(i,j)-w2(i-1,j))/(x(i)-x(i-1))
           go to 502
          elseif(v(i,j).gt.0.d0.and.u(i,j).gt.0.d0) then
              dwdx = (w2(i,j)-w2(i-1,j))/(x(i)-x(i-1))
         go to 502
         else
             go to 52
         endif
502       a(j) =(1./enfhy)*((1.d0/re)*delt*(y(j)-y(j-1)))
         b(j)=1.d0+v(i,j)*delt/(2.*(y(j)-y(j-1)))
     *  + (delt*(1.d0/re))*(y(j+1)-y(j-1))/enfhy
         c(j)=(1./enfhy)*delt*(1.d0/re)*(y(j+1)-y(j))+
     *    v(i,j)*delt/(2.*(y(j)-y(j-1)))
             d(j) =(-u(i,j)*dwdx+(1.d0/re)*dwdxs)*delt/2.
     *                + w2(i,j)+aad*delt/2.*dfidx
52        continue
         if(i.lt.ki ) then
            ll = 1
            lm = 1
            al = 0.d0
            am = 0.d0
           go to  5022
         elseif(i.gt.ki.and.i.lt.ke) then
         ll = 1
          lm = 1
c********* newwmann***************
c         lm = 2
c        al=(-8.*f(i,2)+f(i,3))/(3.5*(y(2)-y(1))**2-(y(2)-y(1))*
c     *   (y(3)-y(2))-(y(3)-y(2))**2/2.)
c*************** 2nd order vort b.c,.***************
         al = (-8.*f(i,2)+f(i,3)+7.*f(i,1))/(3.5*(y(2)-y(1))**2
     *    -(y(2)-y(1))*(y(3)-y(2))-(y(3)-y(2))**2/2.)
     * - 2.*(f(i+1,1)*(x(i)-x(i-1))+f(i-1,1)*(x(i+1)-x(i))
     *     -f(i,1)*(x(i+1)-x(i-1)))/
     *     ((x(i+1)-x(i))*(x(i)-x(i-1))*(x(i+1)-x(i-1)))
c        am =(-8.*f(i,jnm)+f(i,jn-2)+7.*1.0*(y(jnm)-y(jn))-
c     *  1.0*(y(jn-2)-y(jnm)))/
c     *         (3.5*(y(jnm)-y(jn))**2-(y(jnm)-y(jn))
c     *  * (y(jn-2)-y(jnm))-(y(jn-2)-y(jnm))**2/2.)
          am = 0.d0
         go to 5022
         else
            ll = 1
            lm = 1
            al = 0.d0
            am = 0.d0
         endif
5022         call  gtriv(a,b,c,d,jn,e,ft,wt,ll,al,lm,am)
         do 17 kk = 1,jn
           w(i,kk) = wt(kk)
17       continue
14        continue
         do 804 j = 1,jn
```

```
c       w(1,j) = (-8.*f(2,j)+f(3,j)+7.*f(1,j))/(3.5*
c     *   (x(2)-x(1))**2-(x(2)-x(1))*
c     *   (x(3)-x(2))-(x(3)-x(2))**2/2.)
        w(1,j) = 0.d0
c************** shapiro and o'brien ******************************** c
wp = w(in,j)+(u(inm,j)*delt*(w(inm,j)-w(in,j))/xdel)
c        wpp = w(in,j-1)+(u(inm,j)*delt*(w(inm,j-1)-w(in,j-1))/xdel)
c        w(in,j) = wp + (v(inm,j)*delt*(wpp-wp)/xdel)
c******* newmann ****************************************************
        w(in,j) = w(in-2,j)+xddel*(w(in-1,j)-w(in-2,j))
c       w(in,j) = w(inm,j)
c********* zero end vorticity tryout******
c        w(in,j) = 0.d0
804     continue
        do 814 j = 1,jn
         a(j) = 0.
         b(j) = 0.
         c(j) = 0.
         d(j) = 0.
814      continue
c       vosr = vosm1/vosm2
c       write(*,*)' second half vort :', vosr
        kkk = 0
41       continue
         kkk = kkk +1
        write(*,*) kkk
        if(kkk.gt.25) stop
        do 42 i = 1,inm
         do 42 j = 1,jnm
           f1(i,j) = f(i,j)
42      continue
        delt1 = 0.06d0
c       write(*,*) delt1
c       delt1 = 0.04
C***************************** STREAM FIRST HALF ********************
c*************** downstream stream function calculation *********** c
do 224 j = 2,jnm
c       enfhy = ((y(j+1)-y(j))*(y(j)-y(j-1))*(y(j+1)-y(j-1)))
c        a(j) = -2.d0*(y(j)-y(j-1))/enfhy
c          b(j) = -2.d0*(y(j+1)-y(j-1))/enfhy
c            c(j) = -2.d0*(y(j+1)-y(j))/enfhy
c              d(j) = w(in,j)
c224    continue
c       ll = 1
c       lm = 1
c       al = f(in,1)
c       am = f(in,jn)
c                call  gtriv(a,b,c,d,jn,e,ft,ftt,ll,al,lm,am)
c       do 277 kk = 1,jn
c        f(in,kk) = ftt(kk)
c277     continue
c       do 8115 i = 1,in
c       a(i) = 0.
```

```
c       b(i) = 0.
c       c(i) = 0.
c       d(i) = 0.
c8115     continue
        do 24 j = 2,jnm
         do 55 i = 2,inm
        enfhy =0.
        enfhx = 0.
        dfdysq = 0.
        enfhy = ((y(j+1)-y(j))*(y(j)-y(j-1))*(y(j+1)-y(j-1)))
        enfhx = ((x(i+1)-x(i))*(x(i)-x(i-1))*(x(i+1)-x(i-1)))
        dfdysq = 2.*(f(i,j+1)*(y(j)-y(j-1))+f(i,j-1)*
     *   (y(j+1)-y(j))
     *    -f(i,j)*(y(j+1)-y(j-1)))/enfhy
       a(i) =(1./enfhx)*deltl*(x(i)-x(i-1))
        b(i)=1.d0+ (deltl)*(x(i+1)-x(i-1))/enfhx
        c(i)=(1./enfhx)*(deltl*(x(i+1)-x(i)))
              d(i) =(dfdysq+w(i,j))*deltl/2.
     *                + f(i,j)
55      continue
          ll = 1
c         lm = 2
          lm = 1
          al = f(1,j)
c********** linear extrapolation*********************************
          am = f(in-2,j)+xddel*(f(in-1,j)-f(in-2,j))
c         am = 0.d0
c         am = f(in,j)
         call  gtriv(a,b,c,d,in,e,ft,ftt,ll,al,lm,am)
         do 25 kk = 1,in
          f2(kk,j) = ftt(kk)
25       continue
24       continue
        f2(1,1) = 0.d0
         do 805 i = 1,in
          if(i.le.ki) then
              f2(i,1) = 0.d0
               go to 8005
c***     set the 'i'  below to that of start of ice sheet ********
c          elseif(i.eq.ki) then
c               go to 8005
c       if(i.eq.ki) go to 8005
        elseif(i.gt.ki.and.i.le.ke) then
         vf= v(i,1)
         vi= v(i-1,1)
         xf = x(i)
         xi = x(i-1)
         f2(i,1) = f2(i-1,1)- trap(vf,vi,xf,xi)
         go to 8005
         else
          f2(i,1)=  f2(ke,1)
          endif
c8005   f2(i,jn)= 1.0d0 * (y(jn)-y(jnm)) + f2(1,jnm)
```

```
8005     f2(i,jn)= 1.0d0 * (y(jn)-y(jnm)) + f2(i,jnm)
c        f2(i,jn) = 0.
805      continue
         do 815 i = 1,in
         a(i) = 0.
         b(i) = 0.
         c(i) = 0.
         d(i) = 0.
815      continue
C ************************* STREAM SECOND HALF ***************
          do 26 i  = 2,inm
           do 56 j = 2,jnm
         enfhy = 0.
         enfhx = 0.
         dfdxsq = 0.
          enfhy = ((y(j+1)-y(j))*(y(j)-y(j-1))*(y(j+1)-y(j-1)))
         enfhx = ((x(i+1)-x(i))*(x(i)-x(i-1))*(x(i+1)-x(i-1)))
         dfdxsq = 2.*(f2(i+1,j)*(x(i)-x(i-1))+f2(i-1,j)*
     *  (x(i+1)-x(i))
     *    -f2(i,j)*(x(i+1)-x(i-1)))/enfhx
        a(j)=(1./enfhy)*delt1*(y(j)-y(j-1))
         b(j)=1.d0+ (delt1)*(y(j+1)-y(j-1))/enfhy
         c(j)=(1./enfhy)*(delt1*(y(j+1)-y(j)))
               d(j) =(dfdxsq+w(i,j))*delt1/2.
     *                 + f2(i,j)
56        continue
c***    watch the lt. or.equal to*************
         if(i.le.ki ) then
           ll = 1
           lm = 1
           al = 0.d0
           am = 1.0d0 * (y(jn)-y(jnm))+f2(i,jnm)
         go to 566
         elseif(i.gt.ki.and.i.le.ke) then
         ll = 1
c         lm = 1
          lm = 1
          f2(ki,1) = 0.d0
          vf= v(i,1)
          vi= v(i-1,1)
          xf = x(i)
          xi = x(i-1)
          al = f2(i-1,1)- trap(vf,vi,xf,xi)
c         am = 1.0d0 * (y(jn)-y(jnm))+f2(1,jnm)
          am = 1.0d0 * (y(jn)-y(jnm))+f2(i,jnm)
c         am = 0.
          go to 566
         else
          ll = 1
           lm = 1
           al = f2(ke,1)
           am = 1.0d0 * (y(jn)-y(jnm))+f2(i,jnm)
         endif
```

```
566          call   gtriv(a,b,c,d,jn,e,ft,ftt,11,al,lm,am)
          do 27 kk = 1,jn
          f(i,kk) = ftt(kk)
27        continue
26        continue
         do 806 j = 1,jn
c        f(1,j) = f(1,j)
         f(in,j)= f(in-2,j)+xddel*(f(in-1,j)-f(in-2,j))
806      continue
         do 816 j = 1,jn
          a(j)= 0.
          b(j) = 0.
          c(j) = 0.
          d(j) = 0.
816       continue
         do 44 i = 1,inm
          do 44 j = 1,jnm
           azf = dabs(f1(i,j)-f(i,j))
           azm = dmax1(azf,azm)
44       continue
         if(azm.lt.0005) go to 45
         go to 41
45       continue
         azf = 0.
         azm = 0.
         do 90 i = 2,inm
           do 90 j = 6,jnm
           azf1 = dabs(f1(i,j)-f(i,j))
           azmf = dmax1(azmf,azf1)
           azw1 = dabs(w1(i,j)-w(i,j))
           azmw = dmax1(azmw,azw1)
           azfi1 =dabs(fi1(i,j)-fi(i,j))
           azmfi = dmax1(azmfi,azfi1)
90       continue
         do 990 i = 2,inm
         do 990 j = 6,jnm
          azmc = dabs(w1(i,j)-w(i,j))
           if(azmc.eq.azmw)then
             write(*,*) i,j
           else
             go to 990
           endif
990      continue
c        write(*,*) f(in,2),f(inm,2),f(in-2,2)
c        if(k.lt.5) go to 20
c        do 90 i = 2,inm
c          do 90 j=2,jnm
c          azf1 = 100.*dabs((f1(i,j)-f(i,j))/f(i,j))
c          azmf = dmax1(azmf,azf1)
c          azw1 = 100.*dabs((w1(i,j)-w(i,j))/w(i,j))
c          azmw = dmax1(azmw,azw1)
c          azfi1 =100.*dabs((fi1(i,j)-fi(i,j))/fi(i,j))
c          azmfi = dmax1(azmfi,azfi1)
```

```
c90     continue
c       if(k.eq.1.or.mod(k,100).eq.0) then
        write(*,*)                                  k,time,azmf,azmw,azmfi
c       else
c       go to 92
c       endif
92      amall = dmax1(azmf,azmw,azmfi,amall)
        if(amall.lt.cc.or.k.gt.nmax)
     * go to 150
        amall =0.
        azmf = 0.
        azmw = 0.
        azmfi = 0.
        go to 20
150       open(unit=1,status='new',form='unformatted'
     *          ,file='ice_flom.dat')
           write(1)in,jn,time
            write(1) x,y,f,w,fi,ro,u,v
              write(1) delt,chrl,r,uinf
                close(unit=1)
        stop
        end
        include 'blockifm_ad.for'
```

```
        subroutine strfife
*************************************************************************
*
*************************************************************************
        include 'paramifm.for'
        include 'commbal.for'
        double precision x,y,fi,s,f,w,v,u,ro,time
c       write(*,*) 'tinf,salinf,cc'
c       read(*,30)tinf,salinf,cc
c30     format(f10.0)
        do 60 i = 1 , in
         do 60 j= 1,jn
         fi(i,j)= 0.5
c        fi(i,j) = 0.0
          s(i,j) = salinf
           f(i,j)=0.
            w(i,j)= 0.
             v(i,j)= 0.
              u(i,j)=0.
60      continue
        write(*,*) ki
        do 61 i = 1,ki
          fi(i,1) = 1.0
61      continue
        do 62 i = ke+1,in
          fi(i,1) = 1.0
62      continue
          open(unit=1,status='old',form='unformatted',
     *         file='ice_flom.DAT',err=70)
           read(1)iin,ijn,time
           read(1) x,y,f,w,fi,ro,u,v
                close(unit=1)
70      continue
c*********  set zero stream function at base
c***** comment these out for the ice sheet ******
c       do 111 i = 1,in
c            f(i,1) = 0.d0
c111     continue
c       write(*,*)'enter new tinf - old value=',t(2,1)
c       read(*,'(f10.0)')tinf
c       do 200 i=1,in
c       t(i,jn)=tinf
c       do 200 j=2,jn
c200     t(i,j)= tinf
        return
        end
```

```
function trap(vf,vi,xf,xi)
double precision vf,vi,xf,xi,trap
trap = (vf+vi)*(xf-xi)/2.
 return
  end
```

```
      subroutine gtriv(a,b,c,d,md,e,ft,w,ll,al,lm,am)
c     tridiagonal solver for dirichilet or newmann b.c.s only
      dimension a(md),b(md),c(md),d(md),e(md),ft(md),w(md)
      double precision a,b,c,d,e,ft,w,den,al,am
      if(ll.eq.1) e(1) = 0.
      if(ll.eq.1) ft(1) = al
      if(ll.eq.2) e(1) = 1.
      if(ll.eq.2) ft(1) = -al
      mm = md - 1
      do 1 m = 2,mm
      den = b(m) - c(m)*e(m-1)
c     write(*,*) b(m),den
      e(m) = a(m)/den
1     ft(m) = (d(m) + c(m) * ft(m-1))/den
      if(lm.eq.1) w(md) = am
      if(lm.eq.2) w(md) = (ft(mm) + am)/(1.d0 - e(mm))
      do 2 mk = 1,mm
      m = md - mk
2     w(m) = e(m)*w(m+1) + ft(m)
      return
      end
```

```
      block data
*****************************************************************************
*
*****************************************************************************
      include 'paramifm.for'
      parameter(injn = in*jn,injn = injn*7,injn10=injn*10,in9=in*9)
      include 'commbal.for'
      data diffu, znuref,roref / 1.3083d-07,1.075d-06 ,999.84d0/
c     data romax,pr,q/999.9720,8.219,1.894816/
c     data romax,pr,q/999.9720,13.0,1.894816/
      data romax,alph1,tmax,q/999.972d0,9.297173d-06,4.029325d0
     *  ,1.894816d0/
      data f1,w1,fi1,f2,w2,fi2,u1,v1,u2,v2/injn10*0./
      data a,b,c,d,e,ft,wt,fit,ftt/in9*0./
      data f, w,s,fi, ro, u, v / injn7*0./
      data rsf, rsw, rsfi / 3*0./
      end
```

```
parameter ( in = 89, jn = 36)
parameter (inm =in-1, jnm = jn-1)
```

VITA AUCTORIS

1967 Born in Pune, India on June 8.

1982 Completed the Secondary School Certificate from Tamilnadu State Board, through Santhome Hr.Sec.School, Madras, India.

1984 Completed the Higher Secondary School Certificate from Tamilnadu State Board, through Santhome Hr.Sec.School, Madras, India.

1988 Completed the Bachelor of Engineering from Anna University, through the College of Engineering, Guindy, Madras, India.

1990 Currently a candidate for the degree of Master of Applied Science at the University of Windsor